Blender Wiki PDF Manual

About Blender Wiki PDF Manual

This is an unofficial PDF version of the Blender Wiki Manual pages.

I produced this PDF copy for my personal use since I needed it to learn using Blender, and I could not find an up to date alternative to my knowledge. I've read that Blender documentation is released under the Open Content License (http://opencontent.org/opl.shtml). This license is reported at the end of this pages, below. I am trying to stick to this licence, but i'm no license expert: if you feel something is wrong, just let me know why and possibly how to fix.

The short story

I did not modify the content of the wiki pages except for removing the parts typical of a web page, which are not needed in a "book", since you don't have to browse through web links, but scroll pages.

Since Blender developers are incredibly fast, documenters have always hard times to keep up with that speed, but every week there is something new or updated on the wiki manual. Documenters do a great job, but their work goes easily unnoticed because of the high number of the single pages involved: they're hundreds, and Blender developers they're fast!

Discussing this PDF conversion with other users, I explained that my "method" allows me to update the PDF with the actual web pages quite easily so I told that, if needed, it won't be difficult to update that PDF every month. Many people agreed, and that is what i'm trying to do. I hope that one day there will be an automatically generated PDF directly produced from the wiki. Until then, I plan to keep this PDF updated regularly, and possibly other useful sources of knowledge about Blender, found on its official wiki or web site.

The whole story

I needed an offline manual...

I don't have an internet connection at home, where most of my Blender experiment take place, and always needed a help for learning something I didn't know, or a new feature i've read of on BlenderNation, or elsewhere. I always had to wait until tomorrow, or until monday, if you get what I mean. It was frustrating. I followed a small Character Animation Course for beginners, in the city where I live and they used Maya PLE (which it's free to download but does watermarked renderings and does not import/export too much, btw), and although I hated the interface and find it much more uncomfortable than Blender, which I was used to, I had no problem in learning stuff about it because there was a big complete manual and reference, also in the free, limited edition.

I had to build one.

That was two years ago. Recently, I investigated a bit and found out that there could be a method to:

1. retrieve all the links to the web pages of the wiki manual (http://wiki.blender.org/index.php/Manual)
2. download those pages in the right order, with all the images and external links
3. convert them to a single pdf, ps or html file.
4. fix any possible error and weirdness up to my satisfaction.
5. add pdf bookmarks in order to easily browse chapters and topics .

During the last months i tried different tools in order to to this job better and easily, but actually all the job is done with:

- **Firefox** (http://www.mozilla.com/en−US/firefox/): well, i needed a browser, and Firefox is my choice
- **PHP**. In order to have the correct sequence of pages in the PDF book, I needed to collect all the needed wiki web page links (over 250 of them) in the right order. Easily collect links is not difficult, there are many ways and utilities, but I had to be sure the order of the list obtained was the right one.

  In the past months I used **Linky** (http://gemal.dk/mozilla/linky.html), but I ended up not being satisfied: it does a pretty job but always removes duplicate links and I had issues with this behaviour. In fact, *there are* duplicate pages in the current Blender Manual WIki pages, mainly because some topics belong to or are referred from different chapters. So I ended up writing a PHP script, based on a small URL−retrieving script i found on the net (sorry i forgot where, search for "function crawl_page" on google.com) that collects all the link that are needed, leaves duplicate links if I need and I'm happy this way...
- **HTMLdoc** (http://www.htmldoc.org/): this great GPL software can load html files or download web pages, convert them to a single pdf, ps or html file, allowing you to set−up the conversion in many ways. With this software, you have to setup a "book" file, which contains all the needed settings (quality, options etc) and an ordered list of html files, or links to web pages to convert in a single output document.
- Jpdfbookmarks: this is a free CLI and GUI java tool to import/export and edit PDF bookmarks, and it really was the solution for this. It requires a bit of manual work and patience but not too much to be worried :) I also reported some bugs /whishes to the author, and he was kind enough to fix bugs and add features really fast. Great!

The long road towards perfection ;)

When i had the first usable result, it became clear to me that other people could benefit from it, so i decided to share it through a free web service. At that time I got 19 small PDFs, one for each main section of the manual, and then I published the links to the files on the Blender forum (http://www.blender.org/forum/viewtopic.php?t=13461).

There some user asked for a single PDF, which I thought was a good thing, but I was a bit worried of the resulting size, since most of the wiki manual pages have many pictures (both JPG and PNG formats); then another user, Nathan Letwory aka jesterKing, kindly offered a permanent space on a web site he owns, and so I did. I simply joined the 19 small files into a big one, obtaining a huge PDF that was over 1700 pages and about 87 MB in size.

That result was a bit rough, though: all the menus, logos, navigational links were still there, as HTMLdoc decided to keep them as part of the information of the source web pages. Plus, a part of the many (over 1500) images of the wiki were PNGs, part JPGs, part GIFs. PNGs are good for quality because they use a lossless algorithm but also bigger than JPGs. So I improved that document a bit: removing headers, footers, menus, and other unneeded things in a book, inserting te creation date/time, page numbers, some headers, and converting all the images to JPGs (i was worried to preserve the PNG quality, so i converted them using a 100% quality setting in HTMLdoc), and that alone reduced the size to a much smaller 52 MB. In order to remove part of the HTML before passing it to HTMLdoc, i ended up writing a small CLI php script which downloads all the pages, preprocess them a bit, and saves the fixed files in a local folder.

Then I noticed that other parts of the wiki could benefit (to be shared and printed) from being converted to a handy PDF format. They were:

- Script Catalog from http://wiki.blender.org/index.php/Scripts/Catalog
- Reference from http://wiki.blender.org/index.php/Reference
- Release Logs from http://www.blender.org/development/release−logs

and converted them, too, in the same way, and Nathan agreed that they could be interesting and useful, so he hosted them and published the links on his site.

Some user complained about the size, and some problems in the PDFs: no bookmarks, and several broken internal links between pages. I noticed those problems, too, but since there were no alternatives to those PDF for blender users i preferred to share the documents as they were anyway, for the moment.

So I tried to improve them a bit: for the size problem, i used 85% JPG compression (setting in HTMLdoc), and that alone squeezed the size by a rough 50%! And with almost no visible quality loss! I shoud have tried that before...

Then i found a way to overcome the lack of bookmarks: HTMLdoc should automatically generate bookmarks from HTML tags like H1, H2, and so on but unfortunately it did not work with my HTML files. In such a huge wiki work, there are differences organizazion, styles and tags between the over 250 web pages it's build of! And HTMLdoc gives strange result or even crashes if you set it to build bookmarks. Fortunately i tried another solution! Jpdfbookmarks.

Now, the size of the manual is about 25MB, it has full bookmarks and almost no trace of its specific web page design. External links (tutorials, tools, references,) are working, and as in the online wiki, you can click on any image to get its original version. You can print the manual, and you can search it full−text. Good!

Still, there are some (minor) problems:

- internal links are broken: some of them work and some do not. I don't know why. When they work, often they jump to the "previous" page they refer to. Try to advance of one page, in case;
- there are strange symbols here and there: the HTMLdoc version I heve is not UTF−8 compatible, it renders web pages using ISO−8859−x encodings. I use ISO−8859−1 since i think it's the best for english Language. New versions or other tools may improve this aspect, in the future. Is not something that makes text difficult to understand, though.

As soon as i'll find a good and easy solution, i'll fix them!

How i get it as it is now.

The workflow i use now is this:
- Open the Wiki Manual main page (http://wiki.blender.org/index.php/Manual) in Firefox
- Use a PHP script to download all the useful links in that page, in the right order, and save all the links in a txt file.
- Copy all the page titles and sections from that page (as simple text) in a different TXT file, to later create bookmarks for the PDF.
- Execute the php script that downloads locally and fixes all the pages, and creates the settings file for HTMLdoc.
- Run HTMLdoc and wait for it to finish its job. Having also to download and convert images, it takes half an hour, approximately.
- Edit the bookmark TXT aside of the PDF to check and fix wrong page numbers, and so on.
- Apply that bookmark layout to the PDF, with Jpdfbookmarks (CLI)
- Fix some of the PDF bookmarks settings interactively, with Jpdfbookmarks (GUI)
- Send the files to Nathan :)

I could post the php scripts, but I'm not a great programmer, you know :D and PHP it is so easy.

About this file

This is the last result I had: due to PHP preprocessing pages, I got:

- much improved readability, i think. Let me know.
- less pages (stripped all headers, footers, menus): around 1350. Only! :)

- Over 1750 images included.
- size is just around 25 MB
- there are now full PDF bookmarks
- added a few pages before to include blender documentation license and a description of my work.

You should be able to know what's changed since the last release browsing here:
http://wiki.blender.org/index.php/Special:Recentchangeslinked/Manual

Known issues

I feel that many of the issues i know of are related to the complex HTML used in so many pages. I should eventually strip anything that is not needed, and could cause problems.

- Size is huge (well, no more huge but.. still big!) : there are many many big images and they're useful. I could reduce quality setting for JPGs below 85%, if needed but quality will suffer. Compressing the PDF does not change much the size (tried 7z ultra)
- There could be wrong PDF bookmarks: that's my fault since, still assisted by Jpdfbookmarks, i have to make bookmarks and page numbers refer each other :) thell me what's wrong, i'll fix it.
- Many internal links are broken: some of them work and some do not. I don't know why. When they work, often they jump to the "previous" page they refer to. Try to advance of one page, in case.
- There are strange symbols here and there: HTMLdoc is not UTF−8 compatible, it renders web pages using ISO−8859−x encodings. I use ISO−8859−1 since i think it's the best for english Language. New versions or other tools may improve this aspect, in the future. Is not something that makes text difficult to understand, though.

My modifications to wiki pages

The "processing" I'm referring to is, basically, remove graphical page headers, footers, menus, and other minor html tags that could alter the PDF page structure or distract the eye from the real docs text and pics. The basic principle I follow is: I want to keep only the documentation text and pics, and external links if needed, without anything else, and keep the readability as best as I can.

I hope to be able to be able to convert those wiki docs to pdf until there's a better alternative but, as web content can obviously change, something may go wrong. I can't double check all the pages any time. It will surely improve my Blender knowledge :D but there are simply too many pages. If you note some weirdness or broken section, let me know, i'll try to fix it.

Where to find this PDF

Up to now, you should be able to find it here: http://www.letworyinteractive.com/b/category/blender/. If any change should happen, i'll post on major forums/blogs for everyone to know.

Alternatives

To my knowledge. See:

1. http://wiki.blender.org/index.php/Meta/Wiki_to_PDF
2. http://www.blender.it/manuali.html (bottom of the page)
3. http://www.blender3d.org/e−shop/product_info.php?products_id=79
4. http://www.blendernation.com/2006/04/20/download−the−blender−manual/
5. Download the entire html structure on your PC with tools like wget or winhttrack

Obviously, there are other books or tutorials or videos/dvd about Blender you can download or buy. They're

all useful stuff. Here I am referring only to offline versions of the official Blender manual.

How to contact me

I do this in the spare time so i can't spend too much time on it but if you want to suggest me improvements or other ways to do this, or alternatives, please feel free to do so. Here's how: "m.ardito" is the username and the domain is "libero.it". you know how to use them ;). Please don't post the "reconstructed" address, in no web page, blog, mailing list or newsgroup, anywhere. I already have enough spam to deal with! Thanks.

Have fun! Marco Ardito

Open Content License

OpenContent License (OPL)

Version 1.0, July 14, 1998.

This document outlines the principles underlying the OpenContent (OC) movement and may be redistributed provided it remains unaltered. For legal purposes, this document is the license under which OpenContent is made available for use.

The original version of this document may be found at http://opencontent.org/opl.shtml

LICENSE

Terms and Conditions for Copying, Distributing, and Modifying

Items other than copying, distributing, and modifying the Content with which this license was distributed (such as using, etc.) are outside the scope of this license.

1. You may copy and distribute exact replicas of the OpenContent (OC) as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the OC a copy of this License along with the OC. You may at your option charge a fee for the media and/or handling involved in creating a unique copy of the OC for use offline, you may at your option offer instructional support for the OC in exchange for a fee, or you may at your option offer warranty in exchange for a fee. You may not charge a fee for the OC itself. You may not charge a fee for the sole service of providing access to and/or use of the OC via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You may modify your copy or copies of the OpenContent or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified content to carry prominent notices stating that you changed it, the exact nature and content of the changes, and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the OC or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License, unless otherwise permitted under applicable Fair Use law.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the OC, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the OC, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Exceptions are made to this requirement to release modified works free of charge under this license only in compliance with Fair Use law where applicable.

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the OC. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the OC, or by deriving works herefrom, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the OC.

NO WARRANTY

4. BECAUSE THE OPENCONTENT (OC) IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE OC, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE OC "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE OC IS WITH YOU. SHOULD THE OC PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

5. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MIRROR AND/OR REDISTRIBUTE THE OC AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE OC, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Introduction

Interaction in 3D

Scene Management

Modelling

Modifiers & Deformation

Lighting

Materials

Textures

UV Unwrapping & Texturing

Worlds & Backgrounds

Animation Basics

Character Animation & Armatures

Constraints

Advanced Animation

Effects & Physical Simulation

Rendering

Compositing

Video Sequence Editing

Extending Blender

User Manual: Contents | Guidelines | Blender Version **2.48a**

Introduction

Welcome to Blender! The documentation of Blender consists of many parts: this user manual, a reference guide, tutorials, forums, and many other web resources. The first part of this manual will guide you through downloading Blender, installation, and if you elect to download the sources, building an executable file to run on your machine.

Blender has a very unusual interface, highly optimized for 3D graphics production. This might be a bit confusing to a new user, but will prove its strength in the long run. You are highly recommended to read our section on The Interface carefully, both to get familiar with the interface and with the conventions used in the documentation.

What is Blender?

Blender was first conceived in December 1993 and born as a usable product in August 1994 as an integrated application that enables the creation of a broad range of 2D and 3D content. Blender provides a broad spectrum of modeling, texturing, lighting, animation and video post−processing functionality in one package. Through it's open architecture, Blender provides cross−platform interoperability, extensibility, an incredibly small footprint, and a tightly integrated workflow. Blender is one of the most popular Open Source 3D graphics application in the world.

Aimed world−wide at media professionals and artists, Blender can be used to create 3D visualizations, stills as well as broadcast and cinema quality videos, while the incorporation of a real−time 3D engine allows for the creation of 3D interactive content for stand−alone playback.

Originally developed by the company 'Not a Number' (NaN), Blender now is continued as 'Free Software', with the source code available under the GNU GPL license. It now continues development by the Blender Foundation in the Netherlands.

Key Features:

- Fully integrated creation suite, offering a broad range of essential tools for the creation of 3D content, including modeling, uv−mapping, texturing, rigging, skinning, animation, particle and other simulation, scripting, rendering, compositing, post−production, and game creation;
- Cross platform, with OpenGL uniform GUI on all platforms, ready to use for all versions of Windows (98, NT, 2000, XP), Linux, OS X, FreeBSD, Irix, Sun and numerous other operating systems;
- High quality 3D architecture enabling fast and efficient creation work−flow;
- More than 200,000 downloads of each release (users) worldwide;
- User community support by forums for questions, answers, and critique at http://BlenderArtists.org and news services at http://BlenderNation.com;
- Small executable size, easy distribution;

You can download the latest version of Blender here.

Blender's History

In 1988 Ton Roosendaal co−founded the Dutch animation studio *NeoGeo*. NeoGeo quickly became the largest 3D animation studio in the Netherlands and one of the leading animation houses in Europe. NeoGeo

created award−winning productions (European Corporate Video Awards 1993 and 1995) for large corporate clients such as multi−national electronics company Philips. Within NeoGeo Ton was responsible for both art direction and internal software development. After careful deliberation Ton decided that the current in−house 3D tool set for NeoGeo was too old and cumbersome to maintain and upgrade and needed to be rewritten from scratch. In 1995 this rewrite began and was destined to become the 3D software creation suite we all now know as *Blender*. As NeoGeo continued to refine and improve Blender it became apparent to Ton that Blender could be used as a tool for other artists outside of NeoGeo.

In 1998, Ton decided to found a new company called Not a Number (NaN) as a spin−off of NeoGeo to further market and develop Blender. At the core of NaN was a desire to create and distribute a compact, cross platform 3D creation suite for free. At the time this was a revolutionary concept as most commercial modelers cost several thousands of (US) dollars. NaN hoped to bring professional level 3D modeling and animation tools within the reach of the general computing public. NaN's business model involved providing commercial products and services around Blender. In 1999 NaN attended its first Siggraph conference in an effort to more widely promote Blender. Blender's first 1999 Siggraph convention was a huge success and gathered a tremendous amount of interest from both the press and attendees. Blender was a hit and its huge potential confirmed!

On the wings of a successful Siggraph in early 2000, NaN secured financing of â,¬4.5m from venture capitalists. This large inflow of cash enabled NaN to rapidly expand its operations. Soon NaN boasted as many as fifty employees working around the world trying to improve and promote Blender. In the summer of 2000, Blender v2.0 was released. This version of Blender added the integration of a game engine to the 3D suite. By the end of 2000, the number of users registered on the NaN website surpassed 250,000.

Unfortunately, NaN's ambitions and opportunities didn't match the company's capabilities and the market realities of the time. This over−extension resulted in restarting NaN with new investor funding and a smaller company in April 2001. Six months later NaN's first commercial software product, *Blender Publisher* was launched. This product was targeted at the emerging market of interactive web−based 3D media. Due to disappointing sales and the ongoing difficult economic climate, the new investors decided to shut down all NaN operations. The shutdown also included discontinuing the development of Blender. Although there were clearly shortcomings in the then current version of Blender, with a complex internal software architecture, unfinished features and a non−standard way of providing the GUI, with the enthusiastic support from the user community and customers who had purchased Blender Publisher in the past, Ton couldn't justify leaving Blender to disappear into oblivion. Since restarting a company with a sufficiently large team of developers wasn't feasible, in March 2002 Ton Roosendaal founded the non−profit organization *Blender Foundation*.

The Blender Foundation's primary goal was to find a way to continue developing and promoting Blender as a community−based Open Source project. In July 2002, Ton managed to get the NaN investors to agree to a unique Blender Foundation plan to attempt to release Blender as open source. The "Free Blender" campaign sought to raise â,¬100,000 so that the Foundation could buy the rights to the Blender source code and intellectual property rights from the NaN investors and subsequently release Blender to the open source community. With an enthusiastic group of volunteers, among them several ex−NaN employees, a fund raising campaign was launched to "Free Blender." To everyone's surprise and delight the campaign reached the â,¬100,000 goal in only seven short weeks. On Sunday October 13, 2002, Blender was released to the world under the terms of the GNU General Public License (GPL). Blender development continues to this day driven by a team of far−flung, dedicated volunteers from around the world led by Blender's original creator, Ton Roosendaal.

Version/Revision Milestones

Blender's history and road−map

- 1.00 Jan 1995 Blender in development at animation studio NeoGeo

- 1.23 Jan 1998 SGI version published on the web, IrisGL
- 1.30 April 1998 Linux and FreeBSD version, port to OpenGL and X
- 1.3x June 1998 NaN founded
- 1.4x Sept 1998 Sun and Linux Alpha version released
- 1.50 Nov 1998 First Manual published
- 1.60 April 1999 C−key (new features behind a lock, $95), Windows version released
- 1.6x June 1999 BeOS and PPC version released
- 1.80 June 2000 End of C−key, Blender full freeware again
- 2.00 Aug 2000 Interactive 3D and real−time engine
- 2.10 Dec 2000 New engine, physics, and Python
- 2.20 Aug 2001 Character animation system
- 2.21 Oct 2001 Blender Publisher launch
- 2.2x Dec 2001 Mac OSX version
- **13 October 2002 Blender goes Open Source, 1st Blender Conference**
- 2.25 Oct 2002 Blender Publisher becomes freely available
- Tuhopuu1 Oct 2002 The experimental tree of Blender is created, a coder's playground.
- 2.26 Feb 2003 The first true Open Source Blender
- 2.27 May 2003 The second Open Source Blender
- 2.28x July 2003 First of the 2.28x series.
- 2.30 October 2003 Preview release of the 2.3x UI makeover presented at the 2nd Blender Conference.
- 2.31 December 2003 Upgrade to stable 2.3x UI project.
- 2.32 January 2004 Major overhaul of internal rendering capabilities.
- 2.33 April 2004 Game Engine returns, ambient occlusion, new procedural textures
- 2.34 August 2004 Big improvements: particle interactions, LSCM UV mapping, functional YafRay integration, weighted creases in subdivision surfaces, ramp shaders, full OSA, and many many more.
- 2.35 November 2004 Another version full of improvements: object hooks, curve deforms and curve tapers, particle duplicators and much more.
- 2.36 December 2004 A stabilization version, much work behind the scene, normal and displacement mapping improvements
- 2.37 June 2005 A big leap: transformation tools and widgets, softbodies, force fields, deflections, incremental subdivision surfaces, transparent shadows, and multithreaded rendering.
- 2.40 Dec 2005 An even bigger leap: full rework of armature system, shape keys, fur with particles, fluids and rigid bodies.
- 2.41 Jan 2006 Lots of fixes, and some game engine features.
- 2.42 Jul 2006 The **Node** release. Over 50 developers contributed nodes, array modifier, vector blur, new physics engine, rendering, lipsync and, many other features. This was the release following Project Orange
- 2.43 Feb 2007 The **Multi** release: multi−resolution meshes, multi−layer UV textures, multi−layer images and multi−pass rendering and baking, sculpting, retopology, multiple additional matte, distort and filter nodes, modeling and animation improvements, better painting with multiple brushes, fluid particles, proxy objects, sequencer rewrite, and post−production UV texturing. whew! Oh, and a website rewrite. And yes, it still has multi−threaded rendering for multi−core CPUs. With Verse it is multi−user, allowing multiple artists to work on the same scene collaboratively. Lastly, render farms still provide multi−workstation distributed rendering.
- 2.44 May 2007 The **SSS** release: the big news, in addition to two new modifiers and re−awakening the 64−bit OS support, was the addition of subsurface scattering, which simulates light scattering beneath the surface of organic and soft objects.
- 2.45 Sept 2007 Another bugfix release: serious bugfixes, with some performance issues addressed.
- 2.46 May 2008 The **Peach** release was the result of a huge effort of over 70 developers providing enhancements to the core and patches to provide hair and fur, a new particle system, enhanced image browsing, cloth, a seamless and non−intrusive physics cache, rendering improvements in reflections, AO, and render baking; a mesh deform modifier for muscles and such, better animation support via armature tools and drawing, skinning, constraints and a colorful Action Editor, and much more. It was the the release following Project Peach

- 2.48 Aug 2008 Bugfix release
- 2.48 Oct 2008 The **Apricot** release: cool GLSL shaders, lights and GE improvements, snap, sky simulator, shrinkwrap modifier, python editing improvements

About Free Software and the GPL

When one hears about "free software", the first thing that comes to mind might be "no cost". While this is true in most cases, the term "free software" as used by the Free Software Foundation (originators of the GNU Project and creators of the GNU General Public License) is intended to mean "free as in freedom" rather than the "no cost" sense (which is usually referred to as "free as in free beer"). Free software in this sense is software which you are free to use, copy, modify, redistribute, with no limit. Contrast this with the licensing of most commercial software packages, where you are allowed to load the software on a single computer, are allowed to make no copies, and never see the source code. Free software allows incredible freedom to the end user; in addition, since the source code is available universally, there are many more chances for bugs to be caught and fixed.

When a program is licensed under the GNU General Public License (the GPL):

- you have the right to use the program for any purpose;
- you have the right to modify the program, and have access to the source codes;
- you have the right to copy and distribute the program;
- you have the right to improve the program, and release your own versions.

In return for these rights, you have some responsibilities if you distribute a GPL'd program, responsibilities that are designed to protect your freedoms and the freedoms of others:

- You must provide a copy of the GPL with the program, so that the recipient is aware of his rights under the license.
- You must include the source code or make the source code freely available.
- If you modify the code and distribute the modified version, you must license your modifications under the GPL and make the source code of your changes available. (You may not use GPL'd code as part of a proprietary program.)
- You may not restrict the licensing of the program beyond the terms of the GPL. (You may not turn a GPL'd program into a proprietary product.)

For more on the GPL, check the GNU Project Web site. For reference, a copy of the GNU General Public License is included in Volume II.

Getting support − the Blender community

Being freely available from the start, even while closed source, helped a lot in Blender's adoption. A large, stable and active community of users has gathered around Blender since 1998.

The community showed its best in the crucial moment of freeing Blender itself, going Open Source under GNU GPL in late summer 2002.

The community itself is now subdivided into two, widely overlapping sites:

1. The Development Community, centered around the Blender Foundation site. Here you will find the home of the development projects, the Functionality and Documentation Boards, the CVS repository with Blender sources, all documentation sources, and related public discussion forums. Developers coding on Blender itself, Python scripters, documentation writers, and anyone working for Blender

development in general can be found here.
2. The User Community, centered around the independent site BlenderArtists. Here Blender artists, Blender gamemakers and Blender fans gather to show their creations, get feedback on them, and ask for help to get a better insight into Blender's functionality. Blender Tutorials and the Knowledge Base can be found here as well.

These two websites are not the only Blender resources. The Worldwide community has created a lot of independent sites, in local languages or devoted to specialized topics. A constantly updated listing of Blender resources can be found at the above mentioned sites.

For immediate online feedback there are three IRC chat channels permanently open on irc.freenode.net. You can join these with your favorite IRC client.

The IRC channels are #blenderchat for general discussion of blender; #blenderqa for asking questions on Blender usage; and #gameblender for discussion on issues related to game creation with Blenders included game engine. For developers there is also #blendercoders for developers to ask questions and discuss development issues, as well as a meeting each Sunday at ?; #blenderpython for discussion of the python API and script development; #blenderwiki for questions related to editing the wiki

Who uses Blender?

New releases of Blender are downloaded by more than a million people around the world just in the first 10 days of release. This figure spans all platforms (Windows, Linux, and MacOS) and does not include redistribution, which is fully authorized and unrestricted. We estimate there are in excess of two million users. This manual is written to serve the wide array of talented people that use Blender:

- Hobbyist/Student that just wants to explore the world of computer graphics (CG) and 3D animation
- 2−D artist that produces single image art/posters or enhances single images as an image post−processing lab
- 2−D artist or team that produces cartoon/caricature animations for television commercials or shorts (such as â€œThe Magic of Ameliaâ€)
- 3−D artist that works alone or with another person to produce short CG animations, possibly featuring some live action (such as "Suburban Plight").
- 3−D team that produces an animated (100% CG) movie (such as "Elephant's Dream", "Plumiferos").
- 3−D team that works together to produce live action movies that include some CG.

A wide range of age groups, from teenagers to oldsters use Blender, and the user community is fairly evenly divided between novice and professional graphic artists; those occasional users as well as commercial houses. We can divide the 2−D and 3−D teams that produce movies and animations further into individual job categories. Those that use Blender include:

- Director − Defines what each Scene should contain, and the action (animation) that needs to occur within that scene. Defines shots (camera takes) within that scene.
- Modeler − Makes a virtual reality. Specialties include Character, Prop and Landscapes/Stage modelers
- Cameraman, Director of Photography (DP): sets up the camera and its motion, shoots the live action, renders the output frames.
- Material Painter − paints the set, the actors, and anything that moves. If it doesn't move, they paint it anyway.
- Animation and Rigging − makes things hop about using armatures
- Lighting and Color Specialist − Lights the stage and sets, adjusts colors to look good in the light, adds dust and dirt to materials, scenes, and textures.

- Special Purpose talent − Fluids, Motion Capture, Cloth, dust, dirt, fire, explosions, you know, the fun stuff
- Editor − takes all the raw footage from the DP and sequences it into an enjoyable movie. Cuts out unnecessary stuff.

Audience

Therefore, this manual is written for a very broad audience, to answer the question "I want to *do something*; how do I do it using Blender?" all the way to "what is the latest change in the way to sculpt a mesh?"

This manual is a worldwide collaborative effort using time donated to the *cause celeb*. While there may be some lag between key features being implemented and their documentation, we do strive to keep it as up−to−date as possible. We try to keep it narrowly focused on what you, the end user, need to know, and not digress too far off topic, as in discussing the meaning of life.

There are other Blender wiki books that delve deeper into other topic and present Blender from different viewpoints, such as the Tutorials, the Reference Manual, the software itself, and its scripting language. So, if a question is not answered for you in this User Manual, please search the other Blender wiki books. Okay, if you must know, the meaning of life is to *create*, and Blender is excellent at helping you create beautiful imagery.

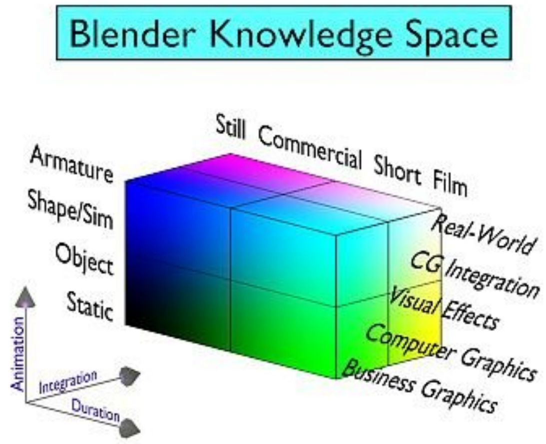
About this Manual

This manual is a mediawiki implementation that is written by a world−wide collaboration of volunteer authors. It is updated daily, and this is the English version. Other language versions are translated, generally, from this English source for the convenience of our world−wide audience. It is constantly out of date, thanks to the tireless work of some 50 or more volunteer developers, working from around the world on this code base. However, it is the constructive goal to provide you with the best possible professional documentation on this incredible package.

To assist you in the best and most efficient way possible, this manual is organized according to the creative process generally followed by 3D artists, with appropriate stops along the way to let you know how to navigate your way in this strange territory with a new and deceptively complex software package. If you read the manual linearly, you will follow the path most artists use in both learning Blender *and* developing fully animated productions:

1. Getting to know Blender = Intro, Navigating in 3d, scene mgt
2. Models = Modelling, Modifiers
3. Lighting
4. Shading = Materials, Textures, Painting, Worlds & Backgrounds
5. Animation = Basics, Characters, Advanced, Effects & Physical Sim
6. Rendering = Rendering, Compositing, Video Seq Edit
7. Beyond Blender = Extending Blender

Learning CG and Blender



Getting to know Blender and learning Computer Graphics (CG) are two different topics. On the one hand, learning what a computer model is, and then learning how to develop one in Blender are two different things to learn. Learning good lighting techniques, and then learning about the different kinds of lamps in Blender are two different topics. The first, or conceptual understanding, is learned by taking secondary and college courses in art and media, by reading books available from the library or bookstore on art and computer graphics, and by trial and error. Even though a book or article may use a different package (like Max or Maya) as its tool, it may still be valuable because it conveys the concept.

Once you have the conceptual knowledge, you can easily learn Blender (or any other CG package). Learning both at the same time is difficult, since you are dealing with two issues. The reason for writing this is to make you aware of this dilemma, and how this manual attempts to address both topics in one wiki book. The conceptual knowledge is usually addressed in a short paragraph or two at the beginning of a topic or chapter, that explains the topic and provides a workflow, or process, for accomplishing the task. The rest of the manual section addresses the specific capabilities and features of Blender. The user manual cannot give you the full conceptual knowledge − that comes from reading books, magazines, tutorials and sometimes a life−time of effort. You can use Blender to produce a full−length feature film, but reading this manual and using Blender won't make you another Steven Spielberg!

At a very high level, using Blender can be thought of as knowing how to accomplish imagery within three dimensions of activity:

1. Integration − rendering computer graphics, working with real−world video, or mixing the two (CGI and VFX)
2. Animation − posing and making things change shape, either manually or using simulation
3. Duration − producing a still image, a short video, a minute−long commercial, a ten minute indie short, or a full−length feature film.

Skills, like navigating in 3D space, modeling, lighting, shading, compositing, and so forth are needed to be productive in any given area within the space. Proficiency in a skill makes you productive. Tools within Blender have applicability within the space as well. For example, the VSE has very little to do with the skill of animation, but is deeply applicable along the Duration and Integration scales. From a skills−learning integration perspective, it is interesting to note that the animation curve, called an Ipo curve, is used in the VSE to animate effects strips.

At the corners/intersections is where most people's interest's lie at any given time; a sort of destination, if you will. For example, there are many talented artists that produce Static−Still−CG images. Tony Mullen's book *Introducing Character Animation With Blender* addresses using CG models deformed by Armatures and shapes to produce a one−minute animation. Using Blender fluids in a TV production/commercial is at the Shape/Sim−Integrated−Minute intersection. Elephants Dream and Big Buck Bunny is a bubble at the Armature−CG−Indie space. Therefore, depending on what you want to do, various tools and topics within

Blender will be of more or less interest to you.

A fourth dimension is Game Design, because it takes all of this knowledge and wraps Gaming around it as well. A game not only has a one−minute cinematic in it, but it also has actual game play, story line programming, etc. −− which may explain why it is so hard to make a game; you have to understand all this stuff before you actually can construct a game. Therefore, this Manual does not address using the Game Engine; that is a whole 'nother wiki book.

Previous: Manual	Contents	*Next:* Manual/Installing Blender

User Manual: Contents | Guidelines | Blender Version **2.4x**

Installing the Binaries

Blender is available both as a binary executable and as source code on the Foundation site (http://www.blender.org/). At the main page click on the 'Downloads' section.

For the online manual hosted at the wiki, you can generally use the most recent version of Blender located at the Blender Foundation website (although all of the features from the newest release version may not be fully updated). If you are using a published version of this manual it is recommended that you use the Blender version included on the Guide CD−ROM. In the following text, whenever "download" is mentioned, those using the book should instead retrieve Blender from the CD−ROM.

Downloading and installing the binary distribution

The binary distributions are provided for 6 primary operating system families (please click your OS for more installation info):

- Windows
- Linux
- MacOSX
- FreeBSD
- Irix
- Solaris

Some unofficial distributions may exist for other operating systems. It's not supported by the Blender Foundation, you should report directly to their maintainers:

- MorphOS

Binaries for the Linux operating systems are provided for two different hardware architectures x86 (Intel and AMD processors) and PowerPC, and the choice between statically linked or dynamically loaded libraries.

The difference between the dynamic and the static binaries is important. The static binaries have the OpenGL libraries compiled in. This makes Blender run on your system without using hardware accelerated graphics. Use the static version if the dynamic version fails! OpenGL is used in Blender for all drawing, including menus and buttons. For this, you will need OpenGL installed on your system. This dependency makes a proper and compliant OpenGL installation at your system a requirement. Generally speaking integrated graphics chips and older low end graphics cards will perform poorly or not at all with Blender due to their poor support for OpenGL. (It is ofter possible to work around the poor OpenGL support of such cards by using software based OpenGL solutions such as by turning down or off hardware acceleration on Windows, or using software MESA 3D on Linux).

Rendering is done by the Blender rendering engine in core memory and by the main CPU of your machine, so an unsupported graphics card will not have an impact if you use the machine only for rendering (as would be the case for a rendering farm).

The installer will create files and several folders in two locations on your computer: one set of folders is for Blender itself, and the other is a user set of folders for your user data. You must have administrator authorization to create these. The folders are:

- .blender − configuration information (mostly prompts in your native language)
- blendcache_.B − temporary space for physics simulation information (softbodies, cloth, fluids)
- plugins − added functionality for textures and sequencing
- scripts − python scripts that extend Blender functionality
- tmp − temporary output, intermediate renders

Python, the Scripting Language

Python is a general purpose scripting language and there is a special interface to access all of Blender's internal functions from that language. Scripts are written in this language that extend the functionality of Blender, without having to re−compile and link the binary distribution. These scripts are written by user−programmers. The recommended version of Python is normally included and installed with the distribution, however you may also download and install it directly from the official Python website, and install it separately. Most functions do not rely on Python; a notable exception is the Help menu which opens a web browser pointed to a specific location. Help text is not bundled into Blender; you must download the latest wiki or pdf user manuals, found here or at www.blender.org.

In general, wherever you install Python, you need to establish an operating system environment variable PYTHONPATH and point it to the Blender Scripts directory where python modules are installed, e.g. "C:\Program Files\Blender Foundation\Blender\scripts\bpymodules" for Windows machines. Environment Variables on Windows machines are set in the advanced Systems settings in the Control Panel.

When Blender starts on a machine with Blender properly installed, you will see this message in the console window:

```
 Compiled with Python version 2.5.
 Checking for installed Python... got it!
```

The above messages means that you have installed Python and have the full development and execution environment, and will be able to access, execute and run all Python scripts that are bundled or available for Blender. If you see a different message, such as:

```
Could not find platform independent libraries <prefix>
Could not find platform dependent libraries <exec_prefix>
Consider setting $PYTHONHOME to <prefix>[:<exec_prefix>]
'import site' failed; use −v for traceback
Checking for installed Python... No installed Python found.
Only built−in modules are available.  Some scripts may not run.
Continuing happily.
```

it just means that the full Python is not available. If you want full Python functionality, refer to the Python site for installation instructions.

When you install Blender, you must tell the Python module where you put the scripts. if you choose to put user data in a different location for each user, then the install will put your scripts in the

```
'C:\Users\<Current User>\AppData\Roaming\Blender Foundation\Blender\.blender\scripts'
```

folder. If you are upgrading, you probably want to overwrite all your old scripts with the new versions, and not have several versions of the same script hanging around on your PC. The best place, if you will not be editing them, is to put them in your Program Files folder with Blender:

1. Do a search on your machine for a file name with the words 'Scripts'.
2. you will see the scripts folder appear after the initial search....C:\Program Files\Blender−2.46/.blender/scripts or something similar....

3. open the script folder from the search window. you will see all the scripts. You can leave em there or put them on your desktop temporarily....
4. Then go to program files, then to blender foundation, then blender folder, then make a new folder called scripts in the blender folder....
5. Drag and drop or copy all the scripts from where ever you put them into this folder.
6. Make sure to include the 2 module folders in the script file.
7. Then, if you don't know this already, Open Blender
8. In Blender, the top menu bar hides all the preferences. Drag it down and then you will see a button marked **file paths**.
9. Once you click that File Paths button a set of path fields will be revealed.
10. Go to the script one and drill down to the script folder you just created in blender where you put all the scripts.
11. Then hit the button that says 'Select Script Paths'.
12. Then go to the file menu and save as default setting so Blender will remember that the script folder is where you told it to look Ctrl U
13. Be careful though if you have already done stuff in blender at this point every time you start it it will be the default start up.

Windows

Quick Install

Download the file `blender-2.##-windows.exe`, (`2.##` being the version number) from the downloads section of the Blender Website. Start the installation by double−clicking the file. This presents you with some questions, for which the defaults should be OK. After setup is complete, you can start Blender right away, or use the entry in the Start menu.

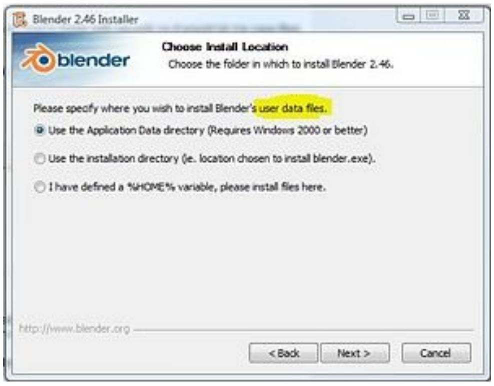
In−depth Instructions

Download the file `blender-2.##-windows.exe` from the 'Downloads' section of the Blender Website. Choose to download it (if prompted), select a location and click "Save". Then navigate to the location you saved the file in and double−click it to start the installation. The first dialog presents you with a helpful reminder: You must be logged on to your PC as ADMINISTRATOR! You will be denied access if you do not have the rights (Vista user especially) to the C:/Program Files folder and/or are authorized to install executable software on the PC. I have Vista install as Administrator (in the file explorer, right−click on the downloaded 'blender−2.46−windows.exe' and select 'Run as administrator' from the right−click menu).

The second dialog presents you with the license. You are expected to accept it if you want the installation to go any further. After accepting the license, select the components you wish to install (there is just one, Blender) and the additional actions:

- Add a shortcut to the Start menu,
- Add Blender's icon to desktop,
- associate .blend files with Blender.

By default they are all checked. If you don't want some action to be taken, simply uncheck it. When done, click on Next. The next dialog is where to put the executable files, usually in the C:\Program Files folder.



The next dialog is tricky, and it is where to put user files. These folders save user data, namely temp data like test renders and physics data. Each user of that PC can have their own, or they call all share one.

Select a place to install the files to (the default should be OK), and click Next to install Blender. Press Close when installation is over.

Afterwards you will be asked whether you want to start Blender immediately. Blender is now installed and can be started by means of the Start menu (an entry named "Blender Foundation" will have been created by the setup routine) or by double−clicking a Blender file (`*.blend`).

After the files are unpacked, Blender will check for required system components, like DLLs, which you must get from Microsoft or your hardware vendor. Most common is a VCRT dll that is/was not bundled with old versions of Windows. After confirmation, you will be able to run Blender!

Portable Install

If, like many people, you are a) obsessed with Blender, and b) have a USB drive, you'll be glad to know that Blender runs beautifully off a USB key. Just download the .zip version and extract it. You may want to avoid having it store the animation output or other temporary files on the drive, as it may shorten the life, but otherwise, Blender runs fine.

OSX

Install

Download the file `blender−2.##−OSX−10.3−py2.#−[platform].zip` from the downloads section of the Blender Website.

- 2.## is the Blender version,
- 10.3 is the minimum OSX version,
- py2.# is the python version, which should be 2.3 for most users, and

- [platform] is either powerpc or i386 (Intel), depending on which type of processor your Mac has.

Python 2.3 is included with the installation. If you wish to use the latest version of Python, please refer to the Python section on this page.

Extract the download by double−clicking the file. This will open a directory with several files.

Mac Users: Since Blender uses OpenGL to render the user interface and Mac OSX draws the entire Desktop with OpenGL as well, you will need to verify that you have sufficient VRAM in your system. Blender will not run with less than 8MB of VRAM. With up to 16 MB VRAM, you will need to set your display to "1000s of colors" (System Preferences −> Displays).

You now can use Blender by double clicking the Blender icon, or drag the Blender icon to the Dock to add its icon there. Blender starts by default in a small window. Hints and tips about the OSX version can be found in the file `OSX tips.rtf` in the installation directory. If Blender doesn't launch, make sure that you downloaded the correct version; oftentimes, newcomers to Blender will accidentally download the Python 2.5 version by accident; try the Python 2.3 version if Blender doesn't seem to launch.

Linux

Quick Install

Download the file `blender-2.##-linux-glibc#.#.#-ARCH.tar.gz` from the 'Downloads' section of the Blender Website. Here `2.##` is the Blender version (currently 2.45), `#.#.#` is the glibc version installed on your computer and `ARCH` is the machine architecture−−either `i386` or `powerpc`. Pick the one matching your system, keeping in mind the difference between static and dynamic builds.

Unpack the archive to a location of your choice. This will create a directory named `blender-2.##-linux-glibc#.#.#-ARCH`, in which you will find the `blender` binary.

To start Blender, open a shell and execute `./blender`, of course while running X.

In−depth Instructions

Download the file `blender-2.##-linux-glibc#.#.#-ARCH.tar.gz` from the 'Downloads' section of the Blender Website. Choose to download it (if prompted), select a location, and click "Save". Then navigate to the location you wish Blender to install to (e.g. `/usr/local/`) and unpack the archive (with `tar -xzf /path/to/blender-2.##-linux-glibc#.#.#-ARCH.tar.gz`). If you like, you can rename the resulting directory from `blender-2.##-linux-glibc#.#.#-ARCH` to something shorter, e.g. just `blender`.

Blender is now installed and can be started on the command line by entering `/path/to/blender` followed by pressing the enter key in a shell. If you are using KDE or Gnome you can start Blender using your file manager of choice by navigating to the blender executable and double−clicking on it.

If you are using the Sawfish window manager, you might want to add a line like `("Blender" (system "blender &"))` to your `.sawfish/rc` file.

To add program icons for Blender in KDE

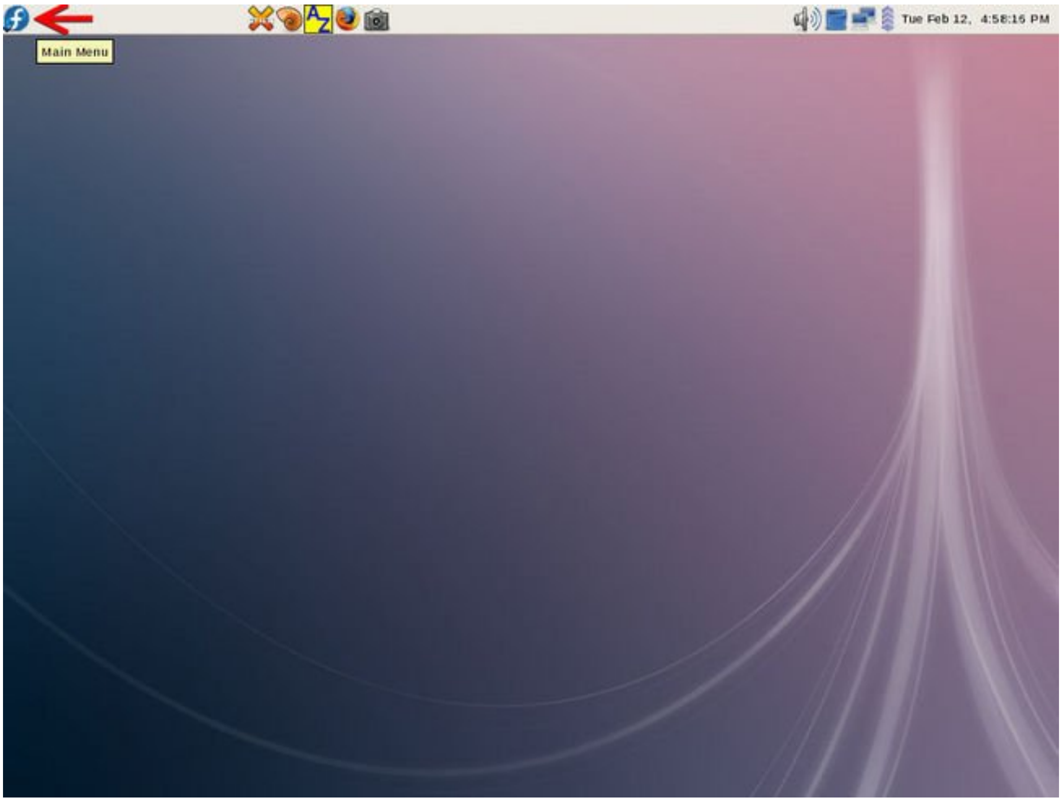
1. Select the Menu Editor from the System sub−menu of the K menu.
2. Select the sub−menu labeled Graphics in the menu list.
3. Click the New Item button. A dialog box will appear that prompts you to create a name. Create and type in a suitable name and click OK.
4. You will be returned to the menu list, and the Graphics sub−menu will expand, with your new entry highlighted. In the right section, make sure the following fields are filled in: Name, Comment, Command, Type and Work Path.
    - The Name field should already be filled in, but you can change it here at any time.
    - Fill the Comment field. This is where you define the tag that appears when you roll over the icon.
    - Click the folder icon at the end of the Command field to browse to the Blender program icon. Select the program icon and click OK to return to the Menu Editor.
    - The type should be Application.
    - The work path should be the same as the Command, with the program name left off. For example, if the command field reads `/home/user/blender−#.##−linux−glibc#.#.#−ARCH/blender`, the Work Path would be `/home/user/blender−#.##−linux−glibc#.#.#−ARCH/`.
5. Click Apply and close the Menu Editor.

To add a link to Blender on the KPanel, RMB click on a blank spot on the KPanel, then hover over Add. Click Button, then Graphics, and select Blender (or whatever you named the menu item in step 3). Alternately, click on the Configure Panel sub−menu in the K menu, click Add, Button, Graphics, and then Blender.

To add a Desktop icon for Blender, open Konquerer (found on the Panel by default, or in the System sub−menu of the K menu) and navigate to the Blender program icon where you first unzipped it. Click and hold the program icon, and drag it from Konquerer to a blank spot on your Desktop. You will be prompted to Copy Here, Move Here or Link Here; choose Link Here.

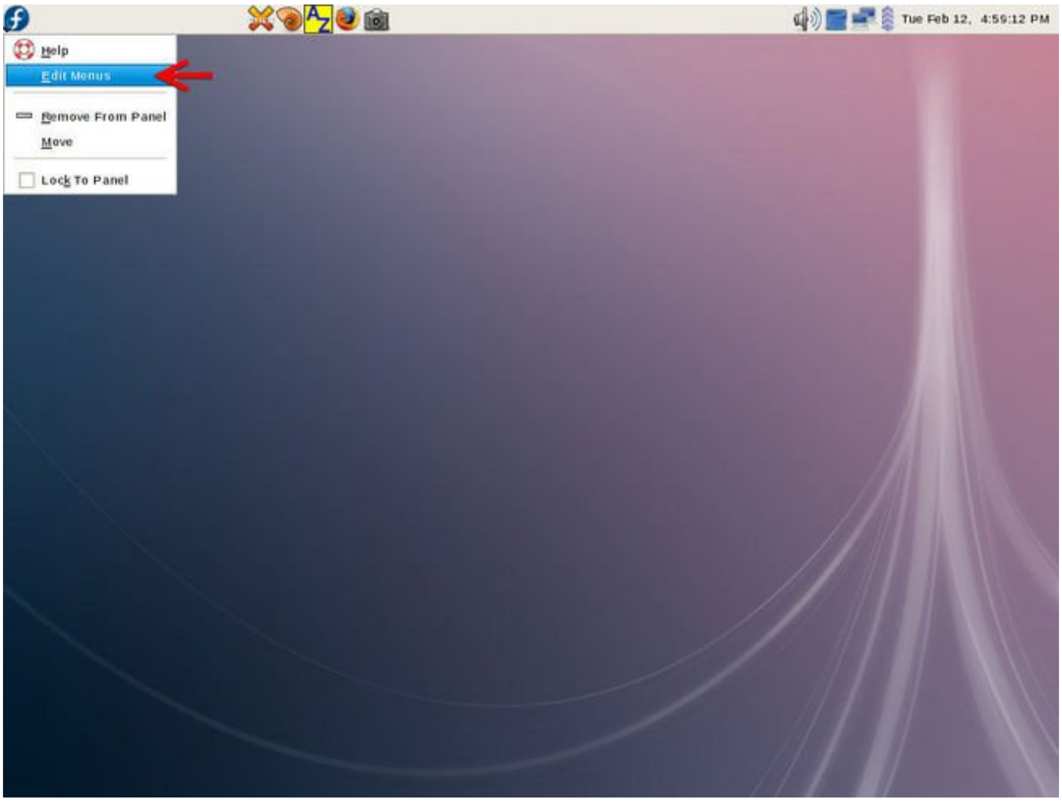
To add program icons for Blender in GNOME

- RMB click the Gnome Main Menu panel (depending on the chosen theme the Icon for the Gnome Main Menu panel could be displayed differently)



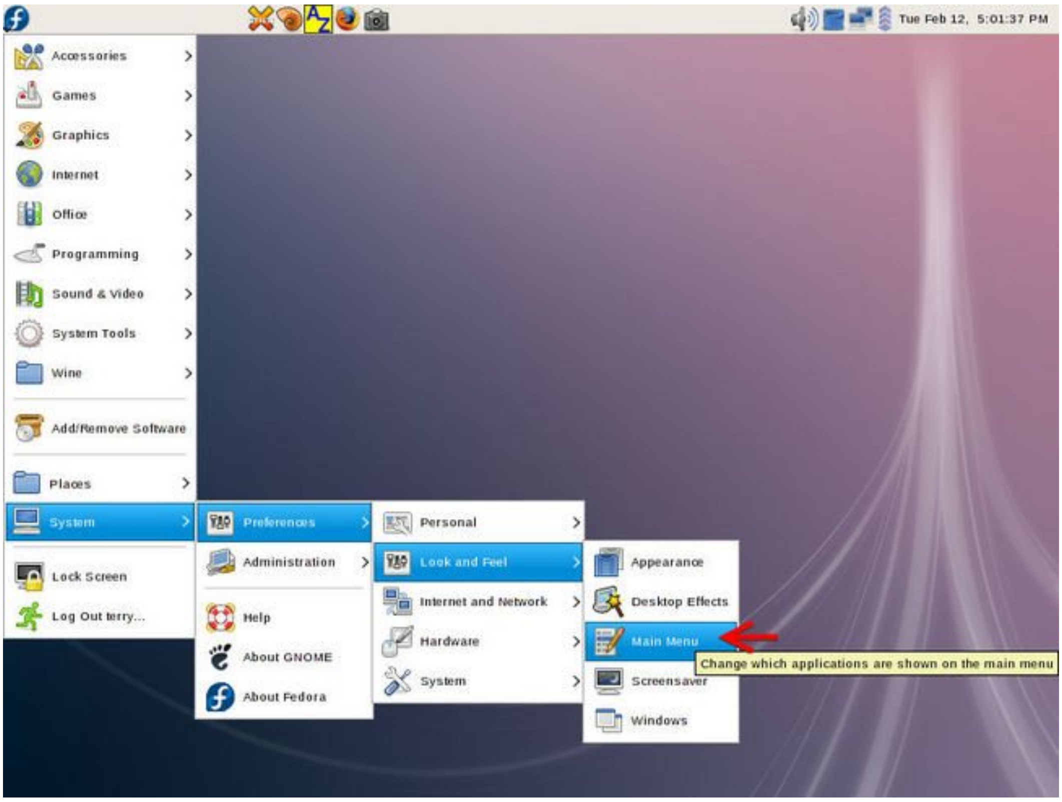
Location of Gnome Main Menu panel

and select Edit Menus from the menu of options that appear,



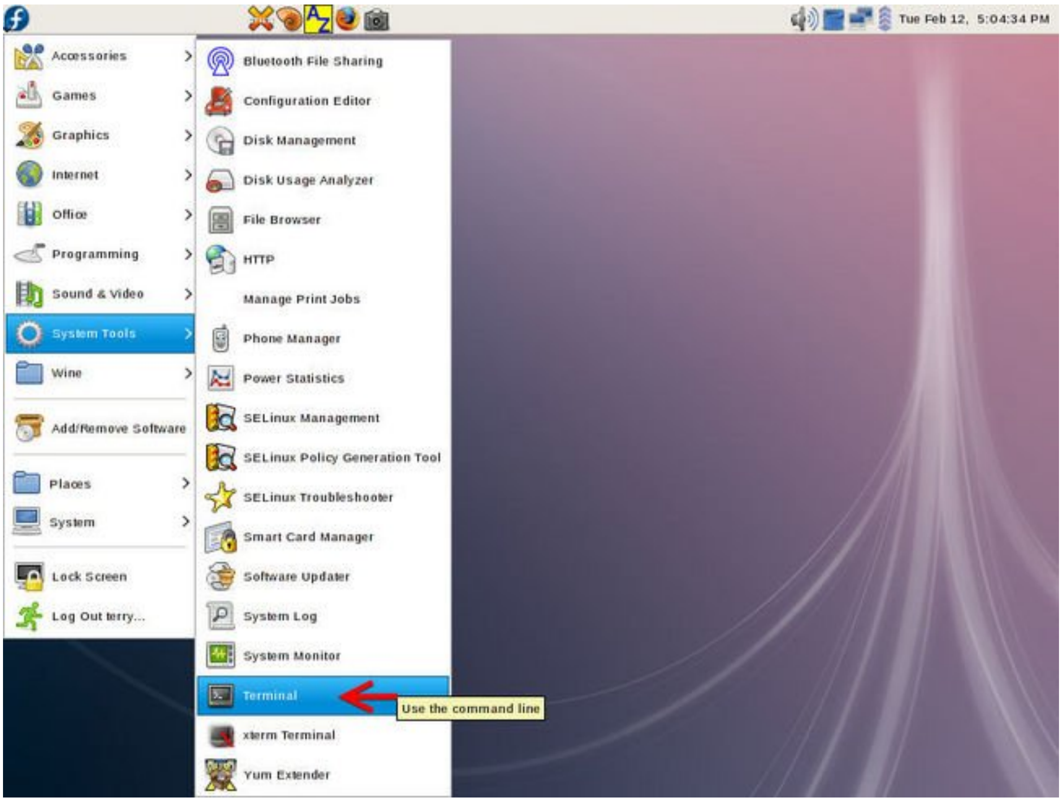
Right clicked panel menu options

or LMB click on the Gnome Main Menu panel and navigate to System > Preferences > Look and Feel > Main Menu (Your menu layout may be different, if so the next option may help).



Location of Main Menu editor

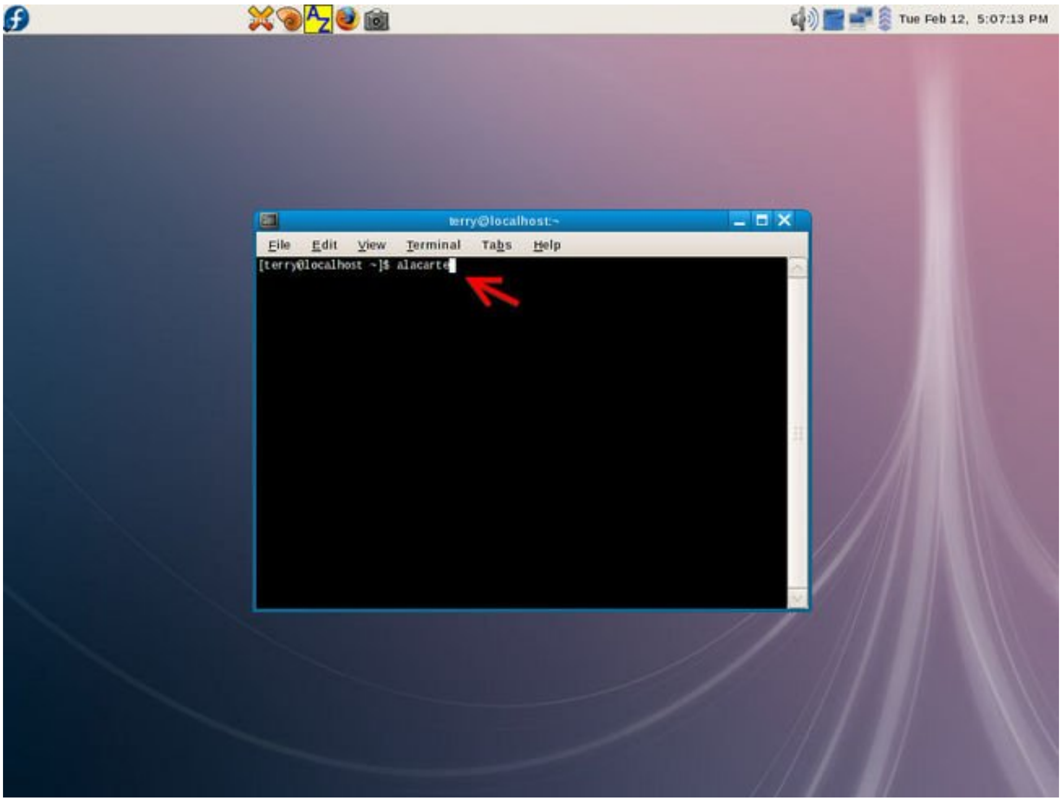
- Yet another method for accessing the Gnome Main Menu editor is to open a Terminal/Console/xterm window



Location of Gnome Terminal

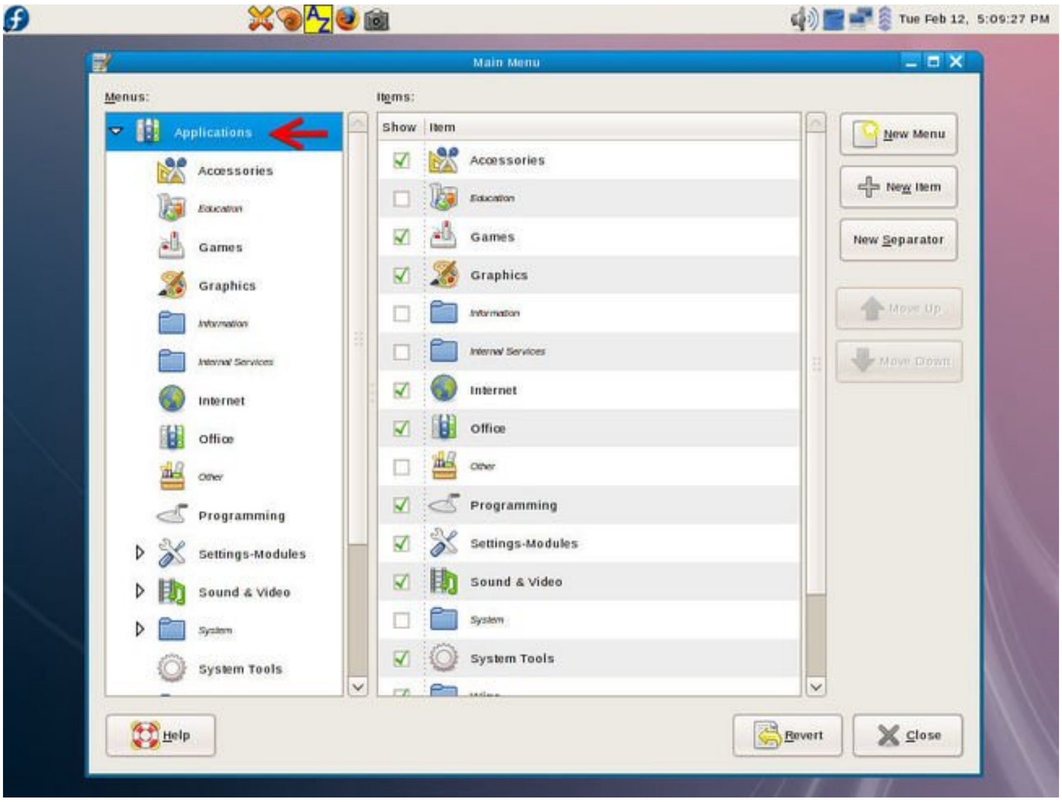
and type the following:

- `alacarte` <PRESS THE ENTER KEY>



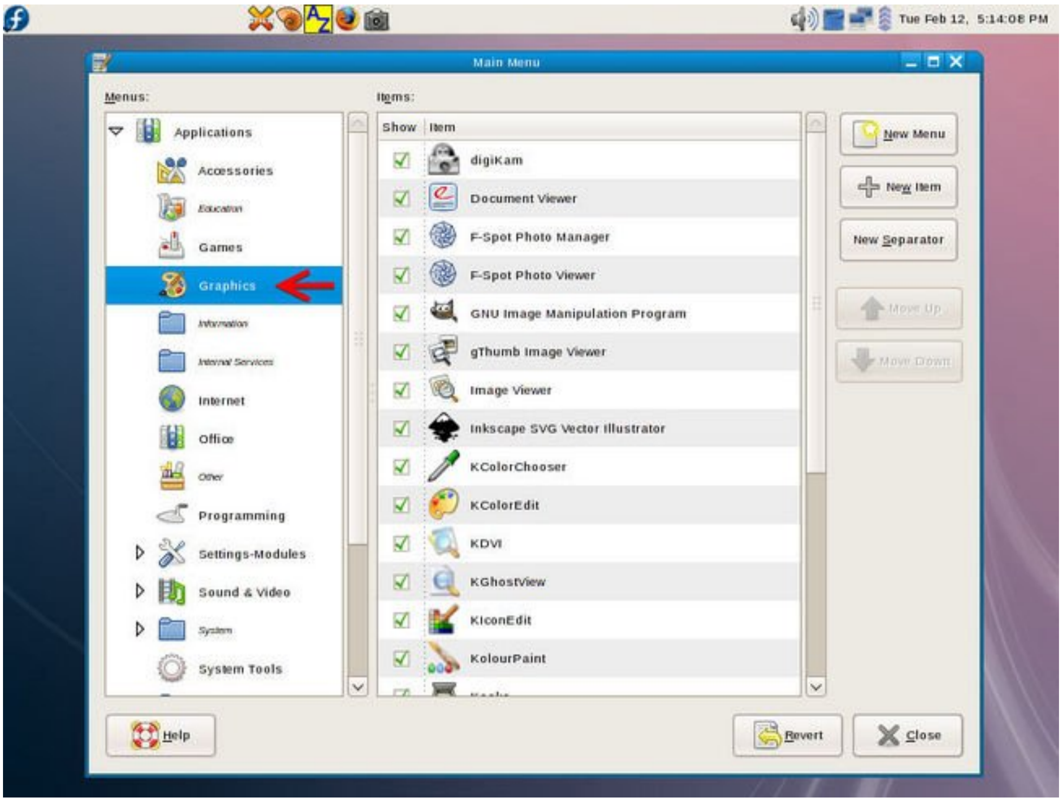
An open Gnome Terminal window

- After using one of the above methods (hopefully) the Main Menu editor is displayed.



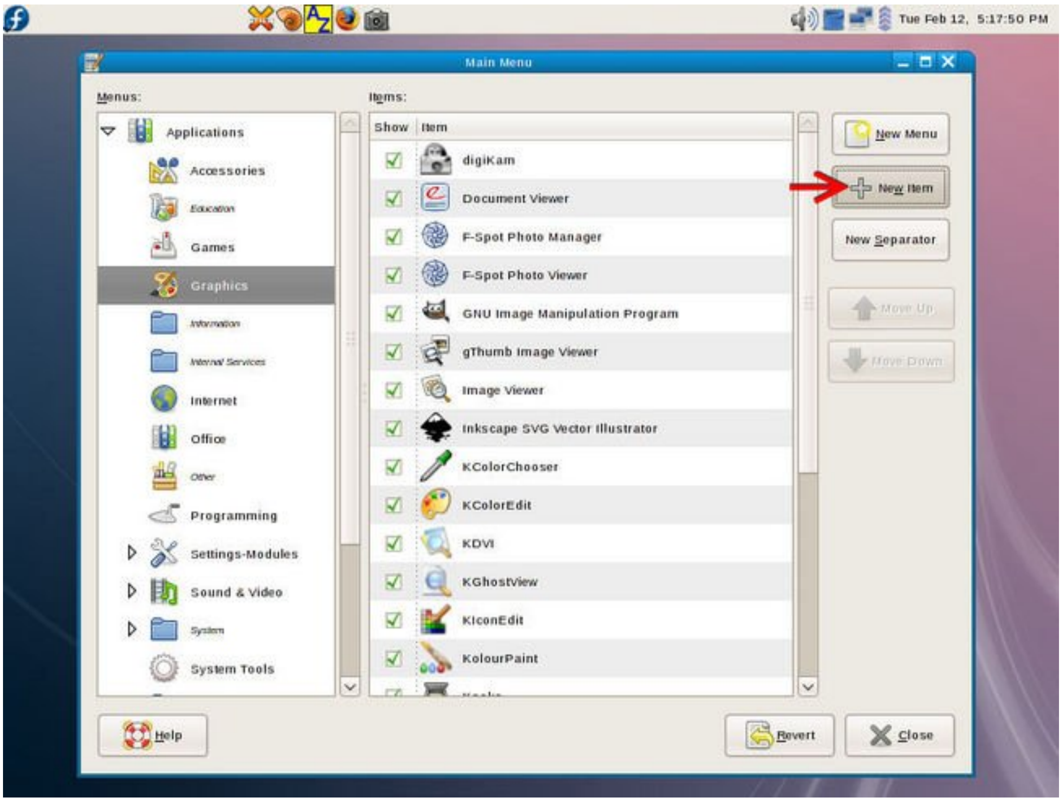
The Main Menu editor windows (alacarte)

- Select the Graphics sub−menu from the Main Menu dialog box (or which ever section you want the Blender icon to be contained in),



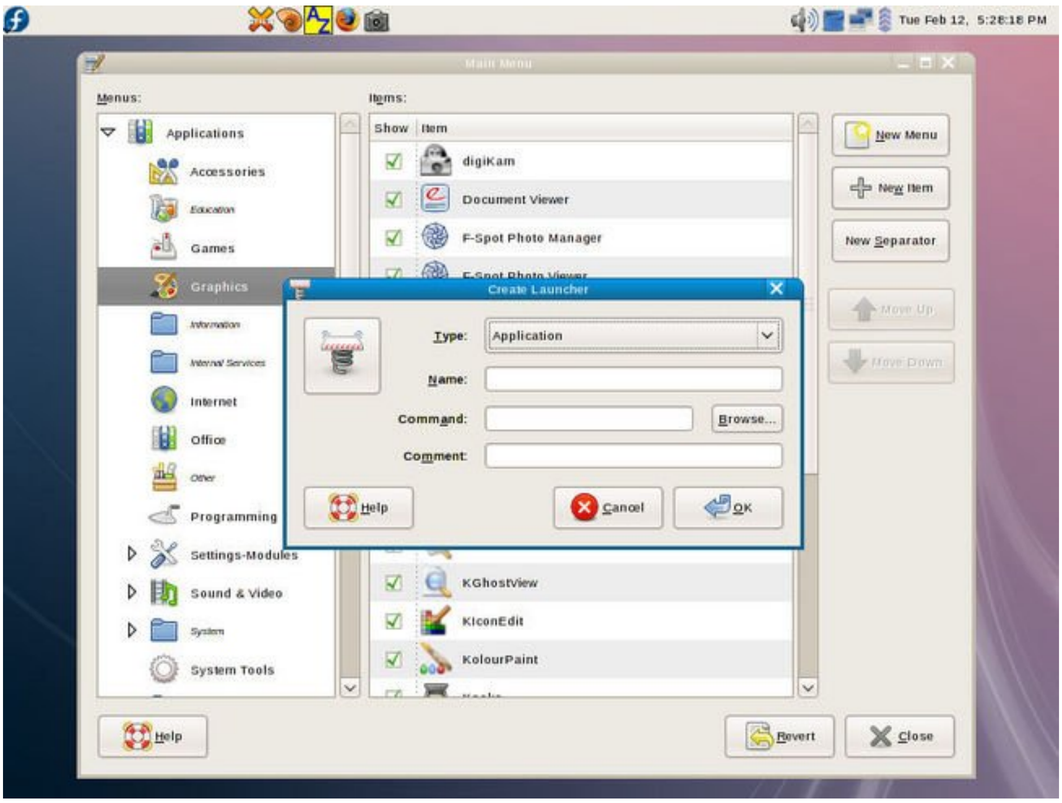
The Main Menu editor windows (alacarte), with the left hand Graphics menu section selected

then click the New Item button.



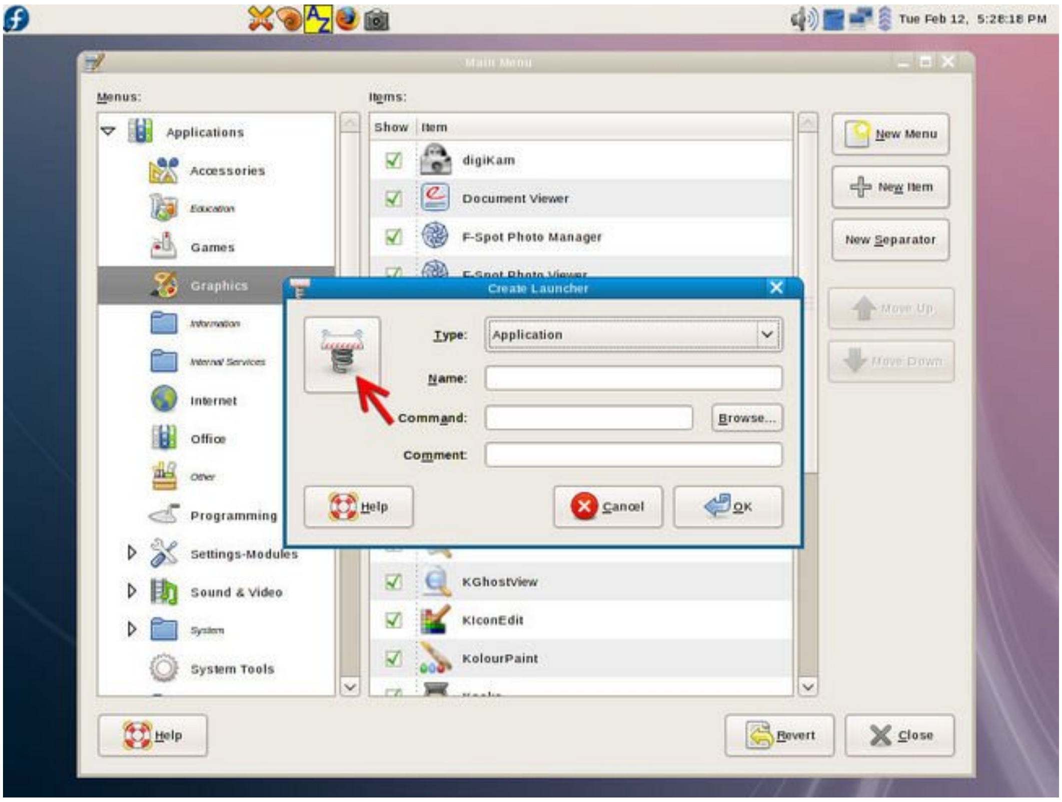
The Main Menu editor windows (alacarte), with the right hand New Item clicked

- In the Create Launcher dialog box, make sure the Type: drop down menu has Application selected.



The Create Launcher dialog box (alacarte)

- In the Create Launcher dialog box also fill in the Name:, Comment: and Command: fields. Fill the Name: field with the program name, for example Blender. You can name this whatever you'd like, this is what appears in the menu, but does not affect the functionality of the program. Fill the Comment: field with a descriptive comment. This is what is shown on the tooltips popups. Fill the Command: field with the full path of the blender program executable/binary, for example, `/home/user/blender-#.##-linux-glibc#.#.#-ARCH/blender`
- Click the icon button to choose an icon (the icon button by default is to the top left within the Create Launcher dialog box and looks like a platform attached to a spring (depending on the chosen theme) or if no icon is selected by the theme the words No Icon may be displayed.



The Icon selection/display box (alacarte)

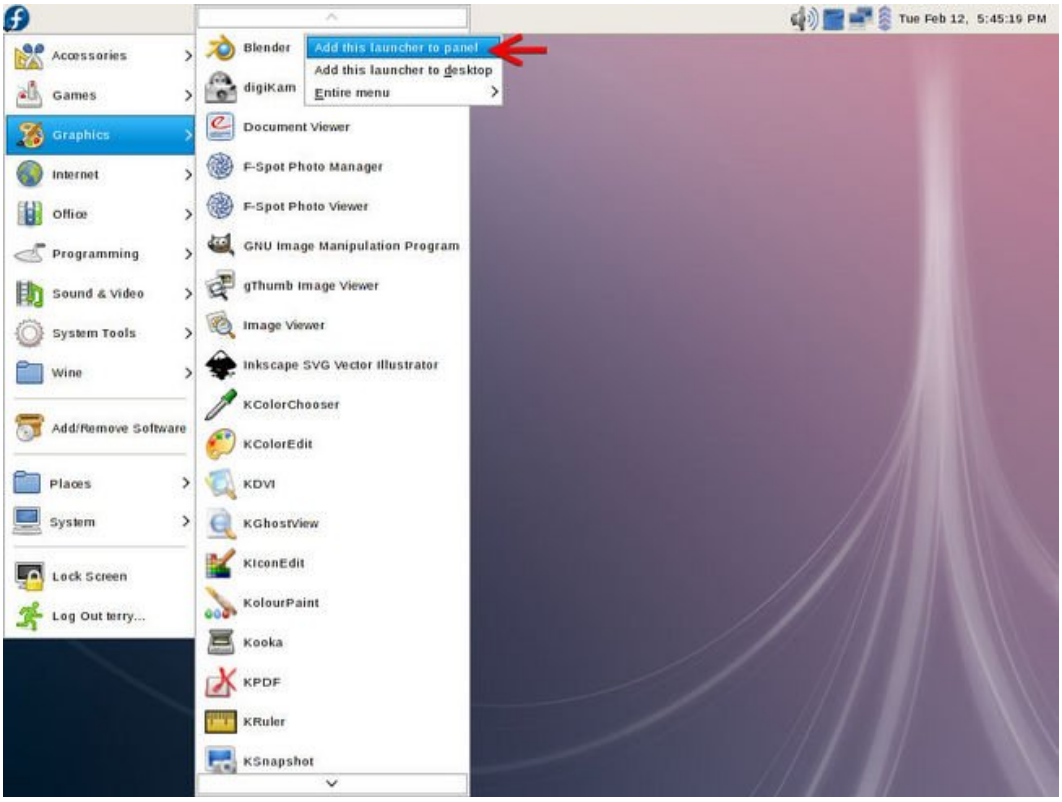
When the mouse is positioned over the icon button it will highlight). There may or may not be an icon for Blender in your default location. You can make one, or look for the icon that goes with KDE. This should be in directory `/opt/kde/share/icons/hicolor/48x48/apps/blender.png` (assuming you installed KDE). If your installation directory is different, you can search for it using this command in a Terminal or Console:

- `find / -name "blender.png" -print` <PRESS THE ENTER KEY>
- If you cannot find your Blender icon then you can use this 2.45 Blender icon for the Gnome Blender icon , just click on the picture and save it to your computer.

- Once you have found the icon you wish to use for Blender, select it in the Browse Icons dialog box and select the Ok button to confirm it.

Browser Dialog displaying available icons in specified location (alacarte)

- Then click the Ok button in the Create Launcher Dialog box to create the new menu and icon item in the Main Menu editor dialog. Make sure the Show item is selected to the left of the newly created Blender entry.
- Click the Close button to close the Main Menu editor.
- Now you should have access to Blender from the Gnome Menu as well as an icon assigned.
- To add a Panel icon for Blender, LMB click on the Gnome Main Menu panel and navigate to the Blender menu entry location in the menu, then RMB click the Blender menu entry and select Add this launcher to panel. Once that is done the Blender icon should appear on the panel.



Right click menu from Main Menu Blender Icon (alacarte)

- To add a Desktop icon for Blender, it is almost the same as adding a Panel icon for Blender but instead of Selecting Add this launcher to panel you instead select Add this launcher to desktop.

FreeBSD

Install

Download the file `blender-2.##-freebsd-#.#-i386.tar.gz` from the downloads section of the Blender Website. Here `2.##` is the Blender version currently 2.45, `#.#` is FreeBSD version and `i386` is the machine architecture.

To start blender just open a shell and execute `./blender`, of course while running X.

Irix

Install

Download the file `blender-2.##-irix-6.5-mips.tar.gz` from the 'Downloads' section of the Blender Website. Here `2.##` is the Blender version (currently 2.45), Python version 2.4, `6.5` is the Irix version and `mips` is the machine architecture.

To start Blender just open a shell and execute `./blender`, of course while running X. Blender was originally developed for the IRIX platform, but is currently not actively being maintained for all IRIX workstation versions. For some workstations performance troubles have been reported.

Solaris

Install

Download the file `blender-2.##-solaris-2.8-sparc.tar.gz` from the 'Downloads' section of the Blender Website. Here `2.##` is the Blender version (currently 2.45), Python version 2.5, `2.8` is the Solaris version and `sparc` is the machine architecture.

Currently no further instructions for Sun Solaris are available. Please use the Blender Website forums for support.

MorphOS

Install

Download the unofficial file `blender_2.##-mos_bin-yyyymmdd.lha` from http://www.yomgui.fr/blender/. Here is `2.##` the Blender version (currently 2.45) and `yyyymmdd` the date stamp. Sources are available on the same link.

Currently no further instructions for MorphOS are available. Please use the Blender Website or MorphOS forums for support.

Configure your Blender

The generic installation of Blender has tons of features and looks pretty cool, too. When you install an upgrade, there are a few things you want to do:

- 'Point' Blender to resources on your machine
- Copy and regression test custom python scripts
- Tell Blender where sequence and texturing plugins are
- Customize your animation, modelling, material, sequence and scripting desktops
- Define your default animation output directory

The top window contains all the User Preferences − click here for more info, including a File Paths tab that you should set up. Then, go into your \Blender\Scripts\ folder and copy the non−distributed scripts into your .blender\scripts directory. Your texture and sequence plugin pathspecs are under User Preferences, and I keep mine in \Blender\bin folders of their own. Your different desktops are selected from the left drop−down menu at the top of the screen. You can size and reconfigure each of these to suit your particular preference (for newbies, the defaults are just fine). If you click the Render button, to top Output directory is where your animations are put (by default), and you might want to point that to your temp directory. Finally, save all your changes with Ctrl U.

Info:

The key combination Ctrl U saves all the settings of the currently open Blender file into the default Blender file (which is usually called .B.blend). The settings in the default Blender file are read when Blender is first started or when Ctrl X is pressed to start a New file. If you accidentally change the settings in your default Blender file there are a few ways of getting back factory default settings:

- Goto the File menu and select Load Factory Settings, once that is done press key combination Ctrl U to save the newly loaded factory settings to the Blender default file. If you have an older version of Blender this method may not be available in that case try the second method.

- Delete the .B.blend file (location can vary between operating systems, check your system) and when Blender is restarted it will recreate it using inbuilt default settings.

Setting up a user directory structure

If you are new to setting up Blender on your PC, you may want to stay organized, as you will quickly accumulate many models, textures, pictures, `.blend` files, `.zip` files, scripts, etc. Mushing them all together in one directory leads to confusion, so it is recommended that you spend a few moments creating a few folders to keep stuff organized. The following is a recommendation based on a few years' experience. There are also free tools to help you manage larger projects (i.e. CVS/Subversion and Verse) but those are beyond the scope of this document.

For casual users, a suggested structure to create on your workstation's hard drive is:

- C:\Blender − a shared folder containing the following subfolders:
  - \bin\ − downloaded binaries (installation exe's) and utilities and add−ons such as Yafray, Python, Gocubic, Panocube, Virtual Dub, etc.
  - \examples\ − work done by others (pictures, movies) for offline study
  - \lib\ − a library of reference material (more on this later)
  - \man\ − User manuals, pdf guides such as Blender Basics, videos from experts, quick reference cards and 'how−to' notes you've made
  - \play\ − your own playground; a directory to save .blend files you're just playing around with
  - \script\ − python scripts that are not distributed with Blender
  - \tmp\ − a temporary place for temporary output; a swap space
  - \tut\ − "how to" tutorials collected from the web. There are many videos and web pages out there (save as a complete web page).
  - \util\ − Blender utilities, such as Make Human, World Forge, and Tree Generator.
  - \work\ − And last but not least, if you actually latch onto a meaningful project that maybe evolves out of the playground, put it here.

Folder explanations

The main Folder is `/Blender/`, which I keep on XP under `/Shared Downloads/`. Create a subfolder `/Blender/bin/` to hold the downloaded binaries or `.exe` install file, as well as any other executable programs associated with Blender, such as Manual/YafRay, and some nifty DLL's you will run across for extending Blender functionality.

- Library: I know you want to create the world, but there are already a bunch of models and stuff out there on the www that other creative people have created. To hold this wealth of pre−built knowledge, create a library (`/Blender/lib/`) to hold this stuff. Subdirectories under that could be `/mesh` (to hold blend files of meshes), `/tex` to hold texture images, and `/pic` to hold pictures, such as reference pictures. My `/blender/lib/mesh` folder has subfolders `/animal`, `/human`, `/machine`, and `/house`, to name a few, holding blend files that contain models of those types of

things. The `/tex` folder has a similar set of folders containing jpg's and even blend files that contain common material settings that are used to color and paint objects. My `/tex` folder contains `/nature`, `/buildings`, `/painted`, and `/metal` subdirectories. The `/pic` folder contains reference pictures of people (Angelina Jolie), faces (my daughter), furniture, my car (a Dodge Viper), and other reference images and concept art that I want to use as reference when modeling.
- Manual and User Guides: Create a `/Blender/man/` file to hold user manuals and guide files in either html, word (`.doc`) and/or `.pdf` formats. There are a few of these floating around. Also, use this folder to save local copies of these wiki pages for off−line reference.
- Tutorials: There are lots of tutorials around and available for downloading. Create a `/Blender/tut/` directory to hold neat tutorials that you find. Some tutorials are hosted by individuals and may disappear, so if you find a tutorial that helps you, download it into this directory.
- Python scripts: Blender uses a scripting language, Python, to extend its functionality. There are dozens of these scripts that can be loaded by Blender. As you find them, save them in a `/Blender/script/` directory, as well as any batch files you write for making backups, etc.
- Utilities: Blender has evolved to the point where there are complete programs that create wondrous things. Keep your Make Human and World Forge utilities (for example) in /Blender/util/.
- Just Do It!: So now YOU need some of your own space, my young padawan. Create `/Blender/play/` and `/Blender/work/` directories to hold play files, and, for when you actually have a meaningful project to work on, a work file. I have used Blender to create a commercial, a documentary on Niger, and a patent (#6,796,205), so I have a subdir under `/Blender/work` for each of those projects. The `/Blender/work/` folder has a folder for each project, and, below that, a set of `/tex`, `/pic`, `/render`, and `/wav` folders to hold textures, pictures, render output, and sound files, respectively. The actual blend files are kept in the `/work/xxx/` folder, where `xxx` is the short name of the project. The `/Blender/play/` folder is loosely organized into `Yafray`, `anim` (animation), `Lighting`, and other folders; basically a trash heap that I rummage around in when I remember that I did something like <insert current problem> but can't remember how to do it again.

Compiling the Source

There are presently four build systems for making a binary for the different operating systems supported. See this web page for more information about compiling a custom installation binary for your machine. This link is in wiki format and provides more information as well.

Compiling the Plugins

Plugins are dynamically loaded routines that augment functionality in either texture generation or sequencing (image manipution). See this thread for more information.

Hardware Support

Blender supports 64−bit hardware platforms running a 64−bit unix operating system, removing the 2Gig addressable memory limit. Work is underway to support a Windows 64−bit OS (call for developer help!)

Blender supports multi−CPU chips, like the Intel Core−Duo and AMD X2 chips by providing a Threads: setting when rendering to work both cores in parallel when rendering an image.

Blender supports a wide variety of pen−based tablets on all major operating systems, in particular OS X, Windows XP, and Linux OSes.

Tips on making Blender run faster and render swifter can be found here.

Previous: Manual/Introduction	Contents	*Next:* Manual/PartI/Interface

User Manual: Contents | Guidelines | Blender Version **2.4x**

The Interface

If you are new to Blender, you should get a good grip on how to work with the user interface before you start modeling. The concepts behind Blender's interface are specifically designed for a graphics modeling application and the vast array of features are different and differently grouped from other 3D software packages. In particular, Windows users will need to get used to the different way that Blender handles controls such as button choices and mouse movements.

This difference is one of Blender's great strengths. Once you understand how to work the Blender way, you will find that you can work exceedingly quickly and productively. Some features are familiar, like the top menu bar of "File", "Add"..."Help". However, many other features are quite unheard of in most (if not all) other applications. For example:

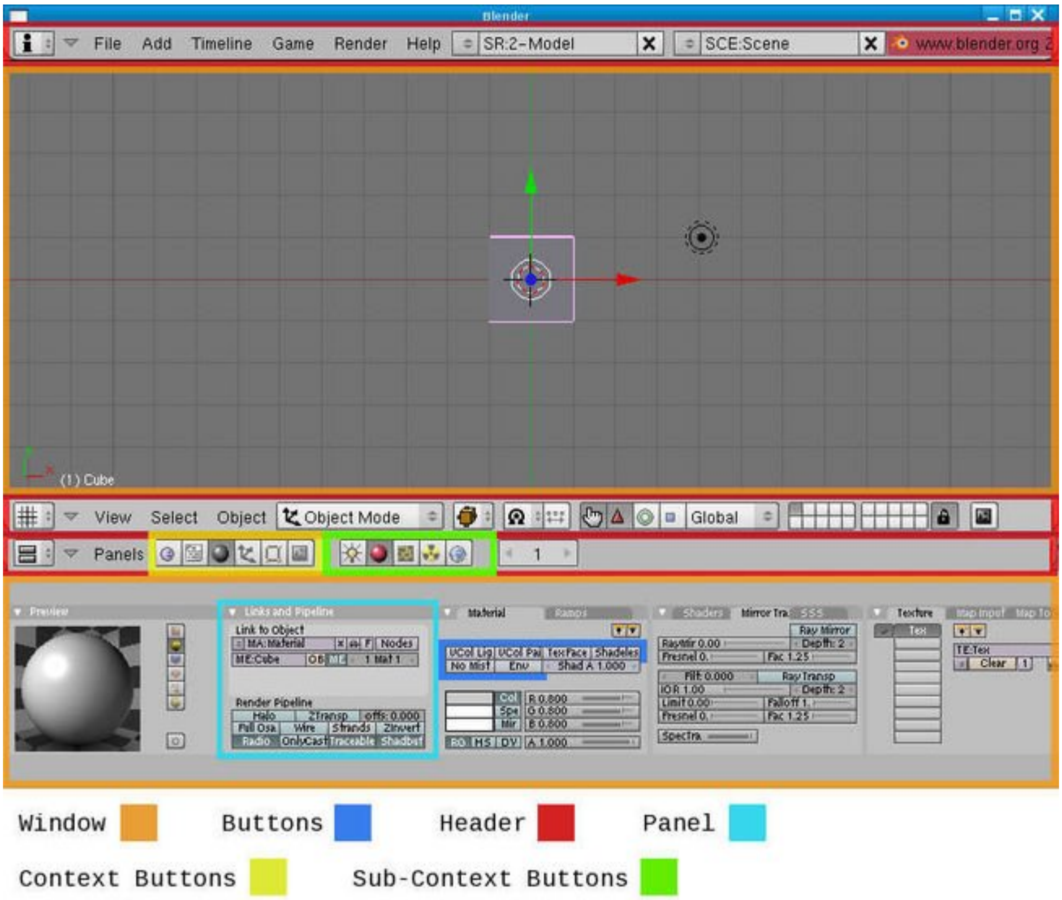
- Blender windows cannot overlap and hide each other, one exception being a small number of mini−floating panels which are transparent, fold−able, small, and dock−able.
- Blender relies heavily on keyboard shortcuts to speed up the work.
- Blender's interface is entirely drawn in OpenGL and every window can be panned, zoomed in/out, and its content moved around.
- Your screen can be organized exactly to your taste for each specialized task and this oganization can be named and memorized.

These key differences (and many others) make Blender a unique, powerful, and very nimble application, once you take the time to understand it.

Blender's Interface Concept

The user interface is the vehicle for two−way interaction between the user and the program. The user communicates with the program via the keyboard and the mouse, and the program gives feedback via the windowing system.

The interface can be broken down into several key areas: *Windows*, *Contexts*, *Panels*, and *Buttons* (controls). For example, The **Button window** contains *Context* buttons which show different groups of *Panels* and the *Panels* each show groups of *Buttons*. These principal areas are discussed on the following pages.



Blender interface organisation

Previous: [Manual/Installing Blender]()	[Contents]()	*Next:* [Manual/PartI/Interface/Keyboard and Mouse]()

User Manual: Contents | Guidelines | Blender Version **2.4x**

Keyboard and mouse

This chapter gives an overview of the general mouse and keyboard usage in Blender and the conventions used in this Manual to describe them, as well as tips on how to use non−standard devices.

Conventions in this Manual

This manual uses the following conventions to describe user input:

- The mouse buttons are called LMB (left mouse button), MMB (middle mouse button) and RMB (right mouse button).
- If your mouse has a wheel, MMB refers to clicking the wheel as if it were a button, while MW means rolling the wheel.
- Hotkey letters are shown in this manual like they appear on a keyboard; for example G which refers to the lowercase "g". When used, the modifier Shift is specified just as the other modifier keys, Ctrl and/or Alt ; this gives, for example, Ctrl W or Shift Alt A.
- NumPad 0 to NumPad 9, NumPad + and so on refer to the keys on the separate numeric keypad. NumLock should generally be switched on.
- Other keys are referred to by their names, such as Esc, Tab, F1 to F12.
- Of special note are the arrow keys, â† , â†’ and so on.

General Usage

Blender's interface is designed to be best used with a three−button mouse. A mouse wheel is quite useful, but not essential.

Because Blender makes such extensive use of both mouse and keyboard, a golden rule has evolved among Blender users: **Keep one hand on the mouse and the other on the keyboard**. If you normally use a keyboard that is significantly different from the English keyboard layout, you may want to think about changing to the English or American layout for your work with Blender. The most frequently used keys are grouped so that they can be reached by the left hand in standard position (index finger on F) on the English keyboard layout. This assumes that you use the mouse with your right hand.

Mouse Button Emulation

It is perfectly possible to use Blender with a two−button mouse or an Apple single−button Mouse. The missing buttons can be emulated with key/mousebutton combos. Activate this functionality in the User Preferences, View and Controls Context, Emulate 3 Button Mouse button.

The following table shows the combos used:

	2−button Mouse	**Apple Mouse**
LMB	LMB	LMB (mouse button)

MMB	Alt LMB	Option LMB (Option/Alt key + mouse button)
RMB	RMB	Command LMB (Command/Apple key + mouse button)

All the Mouse/Keyboard combinations mentioned in the Manual can be expressed with the combos shown in the table. For Example, ShiftAlt RMB becomes ShiftAltCommand LMB on a single−button mouse.

NumPad Emulation

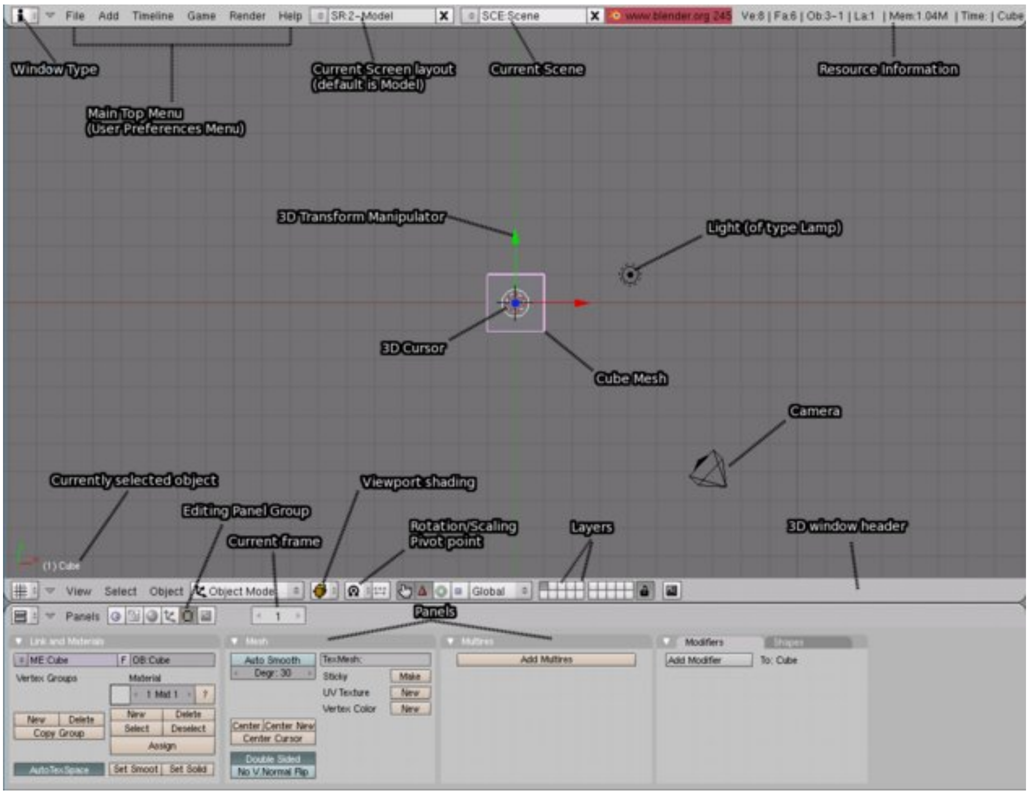
The Numpad keys are used quite often in Blender and are not the same keys as the regular number keys. If you have a keyboard without a Numpad (e.g. on a laptop), you can tell Blender to treat the standard number keys as Numpad keys in the User Preferences, System & OpenGL Context, Emulate Numpad button. A detailed description can be found on this BsoD page.

Previous: Manual/PartI/Interface	Contents	*Next:* Manual/PartI/Interface/Window system

User Manual: Contents | Guidelines | Blender Version **2.4x**

The Window System

When you start Blender you may see a console (text) window open and, shortly after, the main user interface window will display. You may also see a splash screen announcing the Blender version, but it will disappear as soon as you move your mouse.



The default Blender scene.

The default Blender scene shows the screen you should get after starting Blender for the first time.

By default it is separated into three windows:

- The main menu at the top is the header part of a User Preferences window
- A large 3D window (3D Viewport window)
- The Buttons Window (at the bottom)

These windows can be further broken down into separate areas. As an introduction we will cover a few of the basic elements:

- **Window Type**: Allows you to change what kind of window it is. For example, if you want to see the Outliner window you would click and select it.

- **Main Top Menu**: Is the main menu associated with the "User Preferences" window type. To actually see the information, you need to click and drag the area between the 3D window and menu header; Roll the mouse between them and when it changes to a up/down arrow you can drag and see the "User Preferences" window.

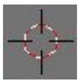
- **Current Screen (default is Model)**: By default, Blender comes with several pre−configured Screens for you to choose from. If you need custom ones, you can create and name them.

- **Current Scene**: Having multiple scenes present allows for you to break up your work into organized patterns.

- **Resource Information** (found in the User Preferences header): Gives you information about application and system resources. It tells you how much memory is being consumed based on the number of vertices, faces and objects in the selected scene. It is a nice visual check to see if you are pushing the limits of your machine.

- **3D Transform Manipulator**: Is a visual aid in transforming objects. Objects can also be transformed (grabbed/moved − rotated − scaled) using the keyboard shortcuts : (g/r/s); Ctrl Space will display the manipulator pop−up. The manipulator visibility can also be toggled by clicking the "hand" icon on the toolbar. The translation/rotation/scale manipulators can be displayed by clicking each of the three icons to the right of the hand icon. Shift LMB −clicking an icon will add/remove each manipulator's visibility.

- **3D Cursor**: Can have multiple functions. For example, it represents where new objects appear when they are first created; Or it can represent where the base of a rotation will take place.
    - Here is the 3D Cursor isolated from the rest of the scene:



- **Cube Mesh**: By default, a new installation of Blender will always start with a Cube Mesh sitting in the center of Global 3D space. After a while, you will most likely want to change the "Default" settings; This is done by configuring Blender as you would want it on startup and then saving it as the "Default" using Ctrl U (Save Default Settings).

- **Light (of type Lamp)**: By default, a new installation of Blender will always start with a Light source positioned somewhere close to the center of Global 3D space.

- **Camera**: By default, a new installation of Blender will always start with a Camera positioned somewhere close to the center of Global 3D space and facing it.

- **Currently selected object**: This field shows the name of the currently selected object.

- **Editing Panel Group**: The bottom window displays panels and those panels are grouped. This row of buttons (called Context Buttons) allows you to select which group of panels are shown. Some buttons will display additional buttons (called Sub−Context Buttons) to the right for selection of sub−groups or groups within groups.

- **Current frame**: Blender is a modeling *and* animation application; As such, you can animate things based on the concept of frames. This field shows what the current frame is.

- **Viewport shading**: Blender renders the 3D window using OpenGL. You can select the type of interactive shading (called Draw Type: in the Blender shading list) that takes place by clicking this button and selecting from a variety of shading styles. You can select from boxes all the way to complex Textured shading. It is recommended that you have a powerful graphics card if you are going to use the Textured style.

- **Rotation/Scaling Pivot point**: Allows you to select where rotation/scaling will occur. For example, rotation could occur about the object's local origin or about the 3D Cursor's position, amongst many

others.

- **Panels**: Help group and organize related buttons and controls. Some panels are visible or invisible depending on what type of object is selected.

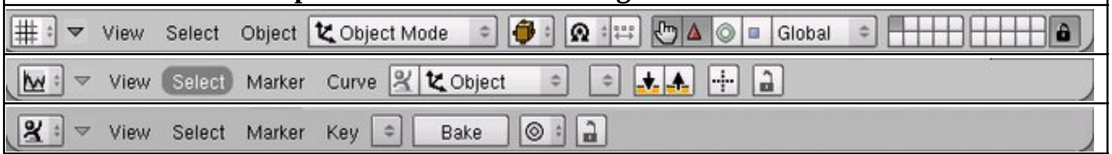
- **Layers**: Make modeling and animating easier. Blender Layers are provided to help distribute your objects into functional regions. For example, one layer many contain a water object and another layer may contain trees, or one layer may contain cameras and lights.

- **3D Window header**: All windows in Blender have a header. This is the header for the 3D window.

The Window Header

Most windows have a header (the strip with a lighter grey background containing icon buttons). We will also refer to the header as the window *ToolBar*. If present, the header may be at the top (as with the Buttons Window) or the bottom (as with the 3D Window) of a window's area.

Sample Window Headers using the Default Theme



Sample Window Headers using the Rounded Theme

If you move the mouse over a window, its header changes to a lighter shade of grey. This means that it is "focused"; All hotkeys you press will now affect the contents of this window.

The icon at the left end of a header, with a click of the LMB, allows selection of one of 16 different window types. Most Window Headers, located immediately next to this first "Window Type" Menu button, exhibit a set of menus. Menus allow you to directly access many features and commands. Menus can be hidden and shown via the triangular button next to them.

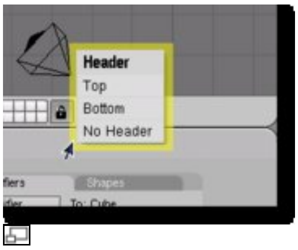
Theme colours:

Blender allows for most of it's interface colour settings to be changed to suit the needs of the user. If you find that the colours you see on screen do not match those mentioned in the Manual then it could be that your default theme has been altered. Creating a new theme or selecting/altering a pre−existing one can be achieved by selecting the User Preferences window and clicking on the Themes section of the window.

The User Preferences window, Theme section selected.

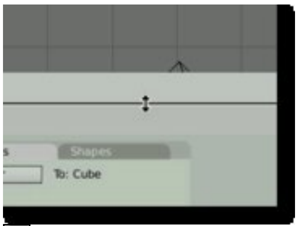
Menus change with Window Type and the selected object and mode. They show only actions which can be performed. All Menu entries show the relevant hotkey shortcut, if any.

There are a few ways to hide a Window Header from a window:

- 

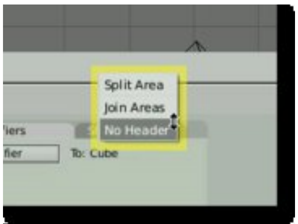
  Screenshot showing the Header popup menu (highlighted in yellow); The result of RMB clicking the Header Window.

  You can hide a particular window's header by moving your mouse over the Window Header that you wish to hide; Then with the mouse cursor still over the Window Header, click RMB to display a popup menu with the name Header; The Header popup menu has the options, Top, Bottom, No Header, select the No Header menu option to hide the Window Header.

- 

  Mouse cursor positioned over the window frame/border showing the UpDown arrow icon.

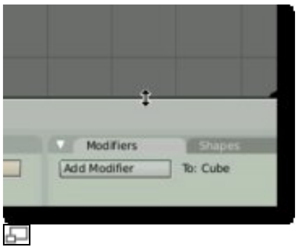
  Another method of hiding a particular window's header is to move your mouse over the dividing frame/border next to the Window Header that you wish to hide (which can be just above or just below the Window Header depending on it's position), when the mouse cursor is positioned correctly it will display as upward and downward pointing arrows;

- 

  Popup menu that results from RMB clicking on the dividing frame/border. With the No Header menu item selected.

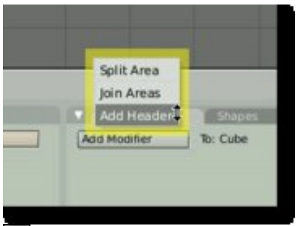
  When the upward and downward pointing arrows are displayed RMB click; A popup menu will be displayed with the options Split Area, Join Areas, No Header, select the No Header menu option to hide the Window Header.

Once a Window Header has been hidden, to redisplay it, do the following:

- 

  Mouse cursor positioned over the Window frame/border of a window with it's header removed.

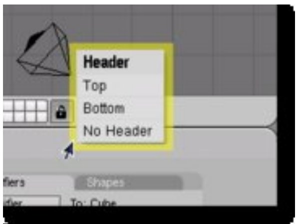
  Move your mouse over the dividing frame/border of the Window Header you wish to unhide (which can be just above or just below the Window Header (that you previously hid) depending on it's position), when the mouse cursor is positioned correctly it will display as upward and downward pointing arrows;

- 

  Popup menu that results from clicking RMB on the dividing frame/border. With the Add Header menu item selected.

  When the upward and downward pointing arrows are displayed click RMB ; A popup menu will be displayed with the options Split Area, Join Areas, Add Header, select the Add Header menu option to add the Window Header back to the window.

- You can also show a hidden header again by clicking the window frame's border with MMB , and selecting Add Header.

- 

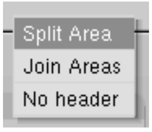
  Screenshot showing the Header popup menu (highlighted in yellow); The result of RMB clicking the Header Window.

  The Window Header can be displayed at the Top or Bottom of the frame. To set a window header's position, RMB click on the Window Header and choose Top or Bottom from the Header popup menu.

Changing Window Frames

You can maximize a window to fill the whole screen with the View Maximize Window menu entry. To return to normal size, use the View Tile Window. A quicker way to achieve this is to use Shift Space, Ctrl or Ctrl to toggle between maximized and framed windows.

You can change the size of a window frame by *focusing* the window you want to split (moving the mouse to its edge), clicking the vertical or horizontal border with MMB or RMB, and selecting Split Area (*The Split menu for creating new windows.*). You can now set the new border's position by moving your mouse to the desired position, and clicking with LMB; or you can cancel your action by pressing Esc. The new window will start as a clone of the window you split. It can then be set to a different window type, or to display the scene from a different point of view (in the case of the 3D Window).



The Split menu

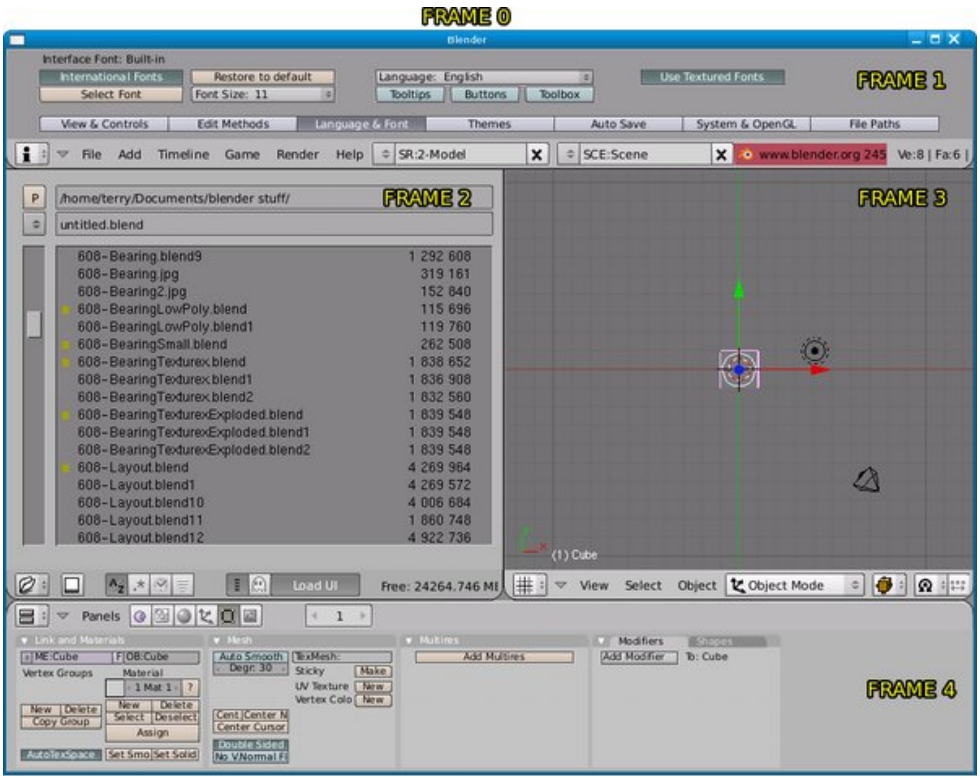
You can resize windows by dragging their borders with LMB.

You can join two windows into one by clicking a border between two windows with MMB or RMB and choosing Join Areas. Then you'll be prompted to click on one of the two windows; the one you click will disappear, while the other will be expanded to cover the full area of both windows. If you press Esc before clicking on one of the windows, the operation will be aborted.

Application Frame (Top Level Frame/Window Manager Frame/Frame 0):

Blender allows the layout of various parts of it's interface to be altered in terms of size and position of it's window frames; However when using window frame actions such as minimizing and maximizing a window frame, all actions are constrained to the current Application Frame dimensions (also known as the Top Level Frame, Window Manager Frame & Frame 0), which is provided by the operating system and is placed around the Blender application as a whole. For example if you currently have your Application Frame only taking up half of your screen and want it to take up all of your screen you would need to click on the outer Application Frame controls for maximizing windows, rather than using one of the possible Blender key combinations such as Ctrl . Using Ctrl while over Frame 2 for example would only make Frame 2 fill the entire space of the Application Frame, not the entire screen (unless the Application Frame was already filling the entire screen). In the screenshot below the Application Frame is indicated by Frame 0 and is light blue with the title Blender in the center of it; Be aware that the Applcation Frame can be different in style, colour and layout and may not be present at all, depending on both the operating system you are running Blender in and the settings used by Blender when it is executed.

Most of the time in this Manual the Application Frame is not shown to both save space and prevent confusion as different operating systems can have different Application Frame layouts.



Blender Application Frame controls.

Interface Items:

Labels in the interface buttons, menu entries, and in general, all text shown on the screen is highlighted in this book like this.

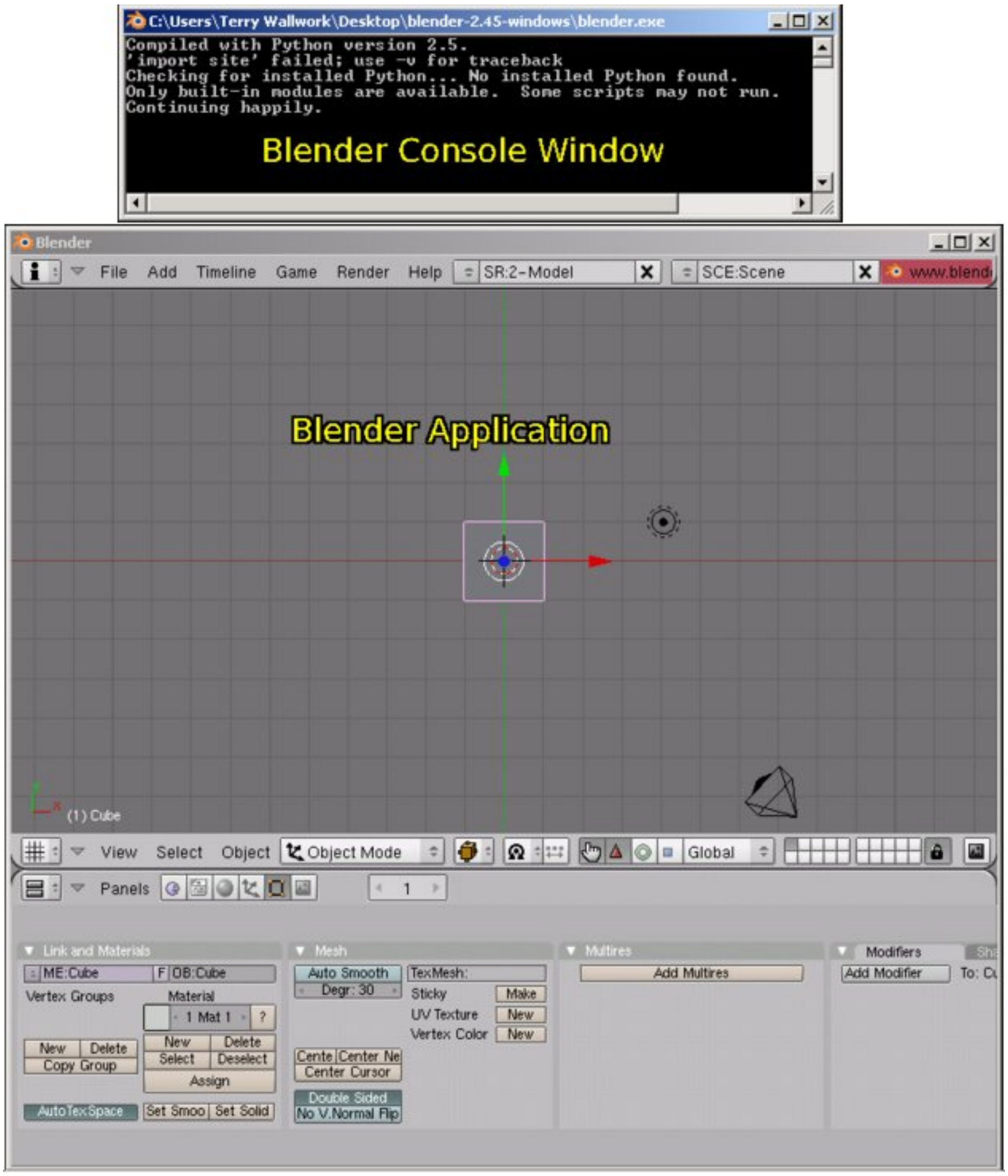
Console Window & Error Messages

The Console Window is an operating system text window that displays messages about Blender operations, status, and internal errors. If Blender crashes on you, it is a good idea to check the Console Window for clues.

Console Window running Windows 2000/Xp/Vista

When Blender is started on a Microsoft Windows OS; The Console Window is first created as a separate window on the desktop; Then assuming the right conditions are met, the main Blender Application window should also appear.

This screenshot shows the 2 windows on a Windows Vista OS:



The Blender Console Window and Blender Application.

The Blender Console Window may not be visible, some reasons for this are:

- The Blender Application window may be covering the Console Window. If this is the case just use the Windows task bar to click on the Blender Console Window icon, which should make the Blender Console Window visible.

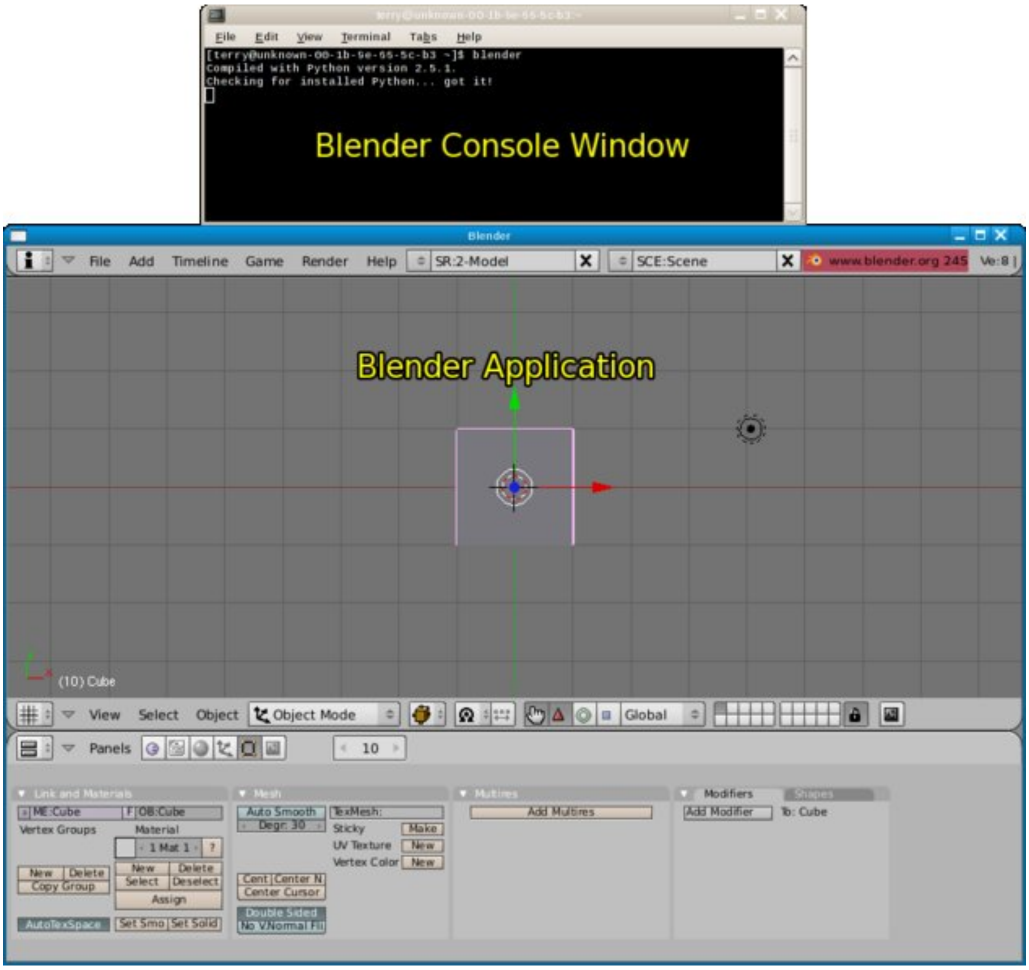
- The Blender Console Window may be minimized/iconifed when Blender starts. If this is the case again, just use the Windows task bar to click on the Blender Console Window icon, which should make the Blender Console Window visible.

Console Window running Linux

The Blender Console Window in Linux will generally only be visible on the Desktop if Blender is started from a Linux Terminal/Console Window, as Blender uses the Console Window it is started in to display it's Blender Console output.

Most of the different Linux distributions have Blender as one of their applications you can install from their packaging systems. When Blender is installed in this way an icon is usually also installed into their menu systems; Allowing for Blender to be started by clicking an icon rather than having to open a separate Linux Console/Terminal window and start Blender from there; When Blender is started using an icon rather than being started from a Terminal window, the Blender Console Window text will most likely be hidden on the Terminal that XWindows was started from.

This screenshot shows a Linux Terminal/Console Window from which Blender is started; Resulting in Blender outputting it's Console text to it:



Blender in Linux started from a Terminal

Closing the Blender Console Window:

The Blender Console Window must remain open while Blender is executing; If the Blender Console Window is closed then the Blender Application window will also close, and any unsaved Blender work will be lost! The MS DOS command windows and Blender Console Window can look similar, so always make sure that you are closing the correct window (or save your work often in Blender, Ctrl W is your friend!)

Console Window Status & Error Messages

The Blender Console Window can display many different types of Status & Error Messages; These can range in level from Trivial (informing the user what Blender is doing, but having no real impact on Blenders ability

to function) to Critical (serious errors which will most likely prevent Blender carrying out a particular task and may even make Blender non−responsive/shutdown completely). The Blender Console Window messages can also originate from many different sources (Internally from within the Blender code, Externally from Python scripts which Blender executes, and from varied types of Plugins, to mention a few).

Here is a list of some of the Blender Console Window messages:

- Compiled with Python version X.Y.

  Blender has support for a scripting language called Python; There are many different versions of Python. When Blender software is compiled (programmers term for building software), it can be compiled to expect a particular version of Python at or above the version reported on the Blender Console Window. So this message reports the minimum version of Python the current version of Blender will use when running.

- Checking for installed Python... got it!

  Blender can use the Python language in 2 different ways depending on how your system is configured. If you have a fully fledged version of Python installed on your system, and it is a version that is able to be used by Blender; Blender will then use the fully fledged version of the Python interpreter. This allows for more features of Python scripts to be used from within Blender.

- Checking for installed Python... No installed Python found.

  If Blender cannot find a fully fledged version of Python on your system or the version it finds is not able to be used; Blender will use an Internal (cut down) version of Python called PyBlender. Even though the Internal version of Python is less feature rich, for the most part it is able to carry out most of the tasks required of it by Blender. If you come across scripts which seem not to work correctly, it may well be that they require a full version of Python to be used successfully; It could also be that the script you're trying to run was written for a different version of Blender/Python. If you wish access to the widest range of Python functionality then there are a few ways to obtain it. One way is to go to http://www.Python.org website and download the version you require. The Windows version of Python comes with a simple to use installation program. In Linux you are likely to have Python fully installed already, but if not you can either compile it and install it manually (often not very easy), or if you're using a common Linux distribution, have your Linux packaging system install and setup Python for you (usually much easier).

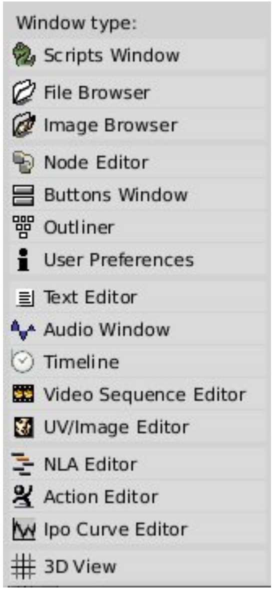
- malloc returns nil()

  When Blender carries out tasks that require extra memory (RAM), it calls a function called malloc (short for memory allocate) which tries to allocate a requested amount of memory to Blender. If however the amount of memory requested by Blender cannot be satisfied malloc will return nil/null/0 to indicate that it failed to carry out a request. If this happens Blender will not be able to carry out the tasks required of it by the user. This will most likely result in Blender shutting down or operating very slowly and non−responsively. If you want to avoid running out of memory; You can either get more memory installed into your system or reduce the amount of detail in your Blender models; Or you can shut down any other programs and services which may be taking up memory that Blender can use.

Previous: Manual/PartI/Interface/Keyboard and Mouse	Contents	*Next:* Manual/PartI/Interface/Window types

User Manual: Contents | Guidelines | Blender Version **2.45**

Window types



The window type selection menu.

The Blender interface, the rectangular window provided by your operating system, is divided up into many rectangular window frames. Each window frame may contain different types of information, depending upon the Window type.

Each window frame operates independently of the others, and you can have the same type of window in many frames. For example, you may have several 3D windows open but each looking at the scene from a different perspective. You can split and merge and resize window frames to suit whatever you are working on. You can also arrange some window frames to show with or without a header to save screen space.

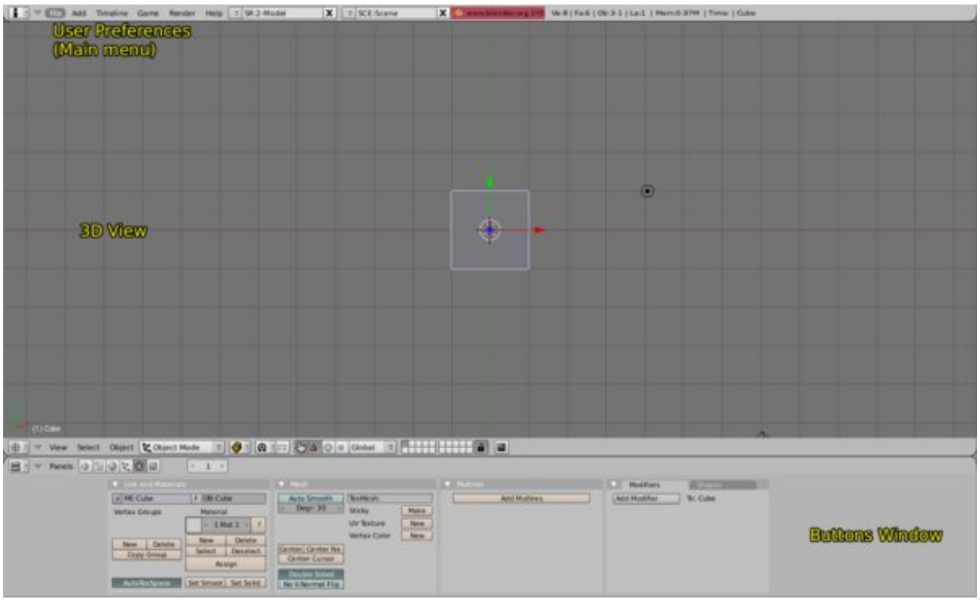
Window types are broken up by functionality:

- Scripts window − user interface for running Python scripts that extend Blender functionality.
- File Browser − for storage and retrieval, especially of .blend files
- Image Browser − search your computer for images, seen as thumbnails
- Node Editor − process/enhance images and materials
- Buttons Window − panels that configure objects and set/select options
- Outliner − Helps you find and organize your objects.
- User Preferences − customize Blender to your work style and computer
- Text Editor − keep notes and documentation about your project, and write Python scripts.
- Audio Window − see sound files and correlate them to frames
- Timeline − Controls for animation playback.
- Video Sequence Editor − assemble video sequences into a filmstrip
- UV/Image Editor − edition of the UVmaps ; edit and paint pictures
- NLA Editor − manage non−linear animation action sÃ©quences.
- Action Editor − combine individual actions into action sequences
- Ipo Curve Editor − manage animation keys and inter/extrapolation of these.
- 3D View − graphical view of your scene

You can select the Window type by clicking the window's header *leftmost* button. A pop−up menu displays showing the available Window types, see (*The Window type selection menu*).

For further details about each Window type, click on it's hyperlink above or visit the reference section III Windows.

The default Blender screen layout is shown below:



Blender default screen layout

Three Window types are provided in Blender's default screen:

3D View

> Provides a graphical view into the scene you are working on. You can view your scene from any angle with a variety of options; see Manual/PartI/Navigating in 3D Space for details. Having several 3D Viewports on the same screen can be useful if you want to watch your changes from different angles at the same time.

Buttons Window

> Contains most tools for editing objects, surfaces, textures, Lights, and much more. You will need this window constantly if you don't know all hotkeys by heart. You might indeed want more than one of these windows, each with a different set of tools.

User Preferences *(Main menu)*

> This window is usually hidden, so that only the menu part is visible − see Manual/PartI/The Vital Functions −> User preferences and Themes for details. It's rarely used though, since it contains global configuration settings. However, the header is frequently used because it provides the only access to a full File menu and to the Add menu.

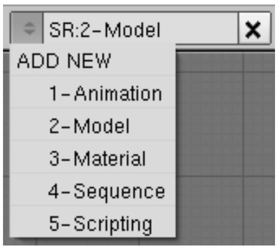
See also

- Reference/Buttons/Window

Previous: Manual/PartI/Interface/Window system	Contents	*Next:* Manual/PartI/Interface/Screens

User Manual: Contents | Guidelines | Blender Version **2.4x**

Screens



Layout dropdown

Blender's flexibility with windows lets you create customized working environments for different tasks, such as modeling, animating, and scripting. It is often useful to quickly switch between different environments within the same file. For each Scene, you need to set the stage by modeling the props, dressing them and painting them through materials, etc. In the example picture in Window system, we are in the modeling stage.

To do each of these major creative steps, Blender has a set of pre−defined screens, or window layouts, that show you the types of windows you need to get the job done quickly and efficiently:

1−Animation
: Making actors and other objects move about.

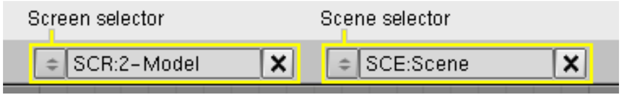
2−Model
: Creating actors, props, and other objects.

3−Material
: Painting and texuring surfaces.

4−Sequence
: Editing scenes into a movie.

5−Scripting
: Documenting your work, and writing custom scripts.

Blender sorts these screen layouts for you automatically in alphabetical and/or numerical order. The preset screen's names typically start with a number. The list is available via the SCR Menu Buttons in the User Preferences Window header shown in (*Screen and Scene selectors*). To change to the next screen alphabetically press Ctrl ; to change to the previous screen alphabetically, press Ctrl .



Screen and Scene selectors

By default, each screen layout 'remembers' the last scene it was used on. Selecting a different layout will switch to the layout **and** jump to that scene.

All changes to windows, as described in Window system and Window types, are saved within one screen. If you change your windows in one screen, other screens won't be affected, but the scene you are working on stays the same in all screens.

Adding a new Screen

As you scroll through the Screen list, you will see that one of the options is to **Add New** − namely, add a new window layout. Click () and select ADD NEW. When you click this, a new frame layout is created based on your current layout.

Give the new screen a *name* that starts with a *number* so that you can predictably scroll to it using the arrow keys. You can rename the layout by LMB into the field and typing a new name, or clicking again to position the cursor in the field to edit. For example you could use the name "6−MyScreen". See (*Screen and Scene selectors*).

Deleting a Screen

You can delete a screen by using the Delete datablock button () and confirm by clicking Delete current screen in the pop−up dialog box. See (*Screen and Scene selectors*).

Rearranging a Screen

Use the window controls to move frame borders. split and consolidate windows. When you have a layout you like, Ctrl U to update your User defaults. The buttons window has a special option, if you RMB on its background, to arrange its panels horizontally across or vertically up and down.

Hints

Overriding Defaults

When you save a .blend file, the screen layouts are saved in it. When you open a file, the LOAD UI button on the file browser header controls whether Blender should use the file's screen layouts, or stick with your current layouts. If LOAD UI is enabled, the file's screen layouts are used, overriding your defaults.

Additional Layouts

With the dramatic increases in functionality, and as you get better at using Blender, based on what you use Blender for, consider adding some other screen layouts (for a complete workflow):

1−Model: 4 3D windows, Buttons window for Editing buttons
2−Lighting: 3D windows for moving lights, UV/Image for displaying Render Result, buttons window for rendering and lamp properties and controls.
3−Material: Buttons window for Material settings, 3D window for selecting objects, Outliner, Library script (if used)
4−UV Layout: UV/Image Editor Window, 3D Window for seaming and unwrapping mesh
5−Painting: UV/Image Editor for texture painting image, 3D window for painting directly on object in UV Face Select mode, 3 mini−3D windows down the side that have background reference pictures set to full strength, Buttons window
6−Animation: Ipo Window, 3D Window for posing armature, NLA Window
7−Node: Big Node Editor window for noodles, UV/Image window linked to Render Result

8−Sequence: Ipo Window, VSE window in Image Preview mode, VSE in timeline mode, a Timeline window, and the good old Buttons window.
9−Notes/Scripting: Outliner, Text Editor (Scripts) window

Reuse your Layouts

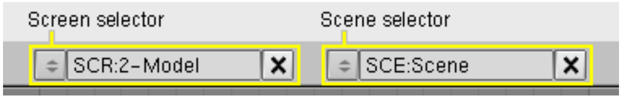
If you create a new window layout and would like to use it for future .blend files, simply save it as a User default by pressing Ctrl U
Delete a layout by clicking the big fat X next to its name, and it is gone for good.

Previous: Manual/PartI/Interface/Window types	Contents	*Next:* Manual/PartI/Interface/Scenes

User Manual: Contents | Guidelines | Blender Version **2.4x**

Scenes

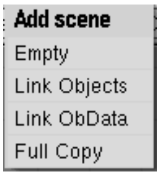
It is also possible to have several scenes within the same Blender file. Scenes may use one another's objects or be completely separate from one another. You can select and create scenes with the SCE menu buttons in the User Preferences Window header (*Screen and Scene selectors*).



Screen and Scene selectors

Adding a new Scene

You can add a new scene by clicking () and selecting ADD NEW. When you create a new scene, you can choose between four options to control its contents (*Add Scene menu*):



Add Scene menu

- Empty creates an empty scene.
- Link Objects creates the new scene with the same contents as the currently selected scene. Changes in one scene will also modify the other.
- Link ObData creates the new scene based on the currently selected scene, with links to the same meshes, materials, and so on. This means that you can change objects' positions and related properties, but modifications to the meshes, materials, and so on will also affect other scenes unless you manually make single−user copies.
- Full Copy creates a fully independent scene with copies of the currently selected scene's contents.

Deleting a Scene

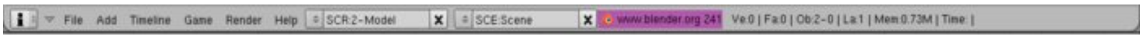
You can delete a scene by using the Delete datablock button () and confirm by clicking Delete current scene to the pop dialog box. See (*Screen and Scene selectors*).

Previous: Manual/PartI/Interface/Screens Contents *Next:* Manual/PartI/Interface/Configuration

User Manual: Contents | Guidelines | Blender Version **2.4x**

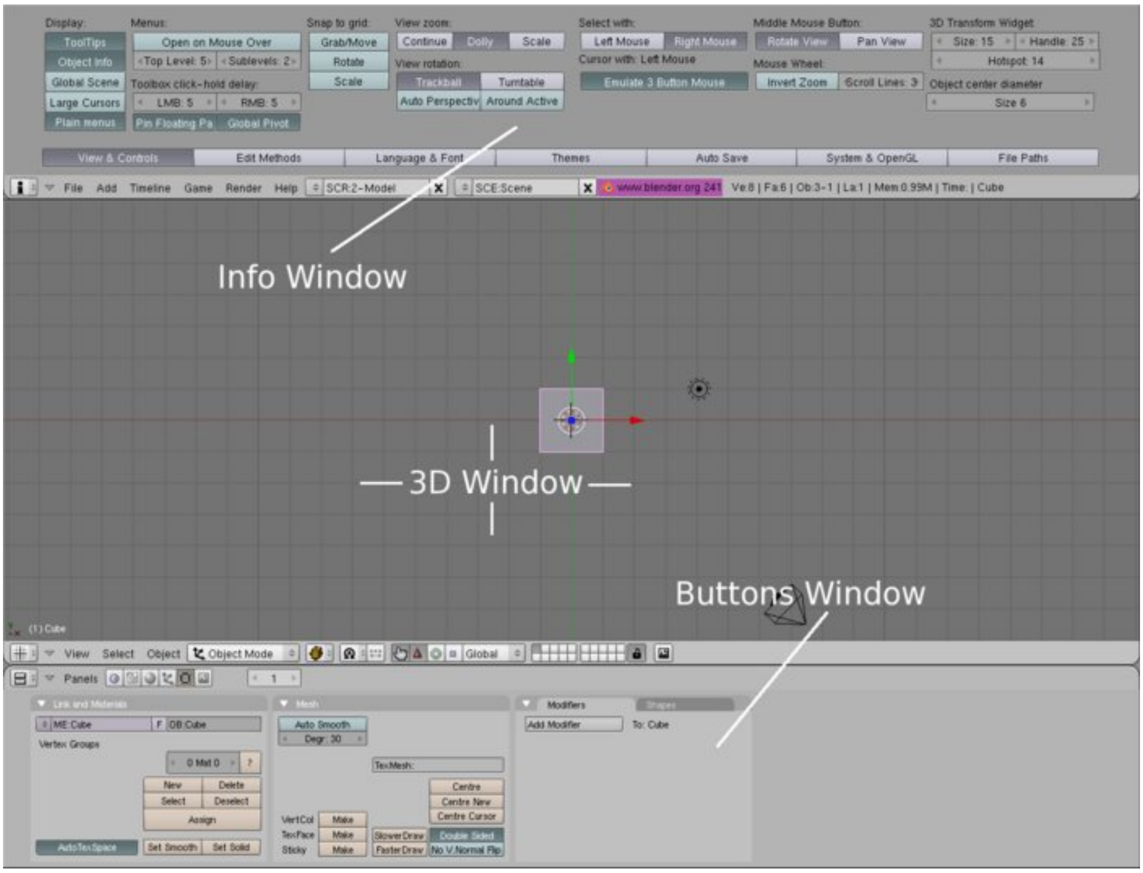
Configuration

The Info window () is where you customize and control Blender. By default this window is located at the top and only the header is visible.



Info window header

To see all of the Info window and its content you need to drag it into view. You can do this by moving the mouse onto the bottom edge of the Info header, or the top of the 3D window, and click the LMB and drag downwards. In picture:Info Visible, the Info window has been made visible at the top.



Info Visible

When viewing all of the Info window you can begin to customize Blender to fit your personality or machine capabilities. For example, you may not like the default theme and switch to the Rounded theme. Or your machine may not be able to handle Vertex Arrays so you switch them off.

For an in depth look at the Info window read the reference section on Info window. There you will find all the details on configuring Blender.

Previous: Manual/PartI/Interface/Scenes Contents *Next:* Manual/PartI/Interface/Contexts

User Manual: Contents | Guidelines | Blender Version **2.4**

Contexts

The Button Window shows six main Contexts, which can be chosen via the first icon row in the header (*Contexts and Sub−Contexts Example*). Each of these might be subdivided into a variable number of sub−contexts, which can be chosen via the second icon row in the header (*Contexts and Sub−Contexts Example*), or cycled through by pressing the same Context button again:

Contexts and Sub−Contexts Example

- Logic (F4) − Switches to Logic context.
- Script − No shortcut. Switches to Script context.
- Shading (F5) − Switches to Shading context.
    - Lamp − No shortcut.
    - Material − No shortcut.
    - Texture − Shortcut F6.
    - Radiosity − No shortcut.
    - World − Shortcut F8.
- Object (F7) − Switches to Object context.
    - Object − No shortcut.
    - Physics − No shortcut.
- Editing (F9) − Switches to Editing context.
- Scene (F10) − Switches to Scene context.
    - Rendering − No shortcut.
    - Anim/Playback − No shortcut.
    - Sound − No shortcut.

Once the Contexts is selected by the user, the sub−context is usually determined by Blender on the basis of the active Object. For example, with the Shading context, if a Lamp Object is selected then the sub−context shows Lamp Buttons. If a Mesh or other renderable Object is selected, then Material Buttons is the active sub−context, and if a Camera is selected the active sub−context is World.

The Buttons in each context are grouped into Panels.

The menu of available options, shown in a window's header, may change depending on the mode of that window. For example, in the 3D View window, the Object menu in Object mode changes to a Mesh operations menu in Edit mode, and a paint menu in Vertex Paint mode. If you are reading this manual and some menu option is referenced that does not appear on your screen, it may be that you are not in the proper mode, or context, for that option to be valid.

Previous: Manual/PartI/Interface/Configuration	Contents	*Next:* Manual/PartI/Interface/Menus

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.4x**

Menus

Top Level

Blender contains many menus each of which is accessible from either the window headers or directly at the mouse's location using HotKeys, see Toolbox.

For example, you can access the Toolbox in the 3D window using either the mouse or the keyboard. From the keyboard you would use the SPACE. To access it using the mouse just hold down the LMB or RMB buttons for a few seconds and the Toolbox will pop−up. (*Top Level*) is the top most menu of the Toolbox.

Some menus are context sensitive in that they are only available under certain situations. For example, the Booleans menu is only available in Object Mode using the (W) hotkey. The same hotkey (W) in Edit Mode brings up the Specials menu.

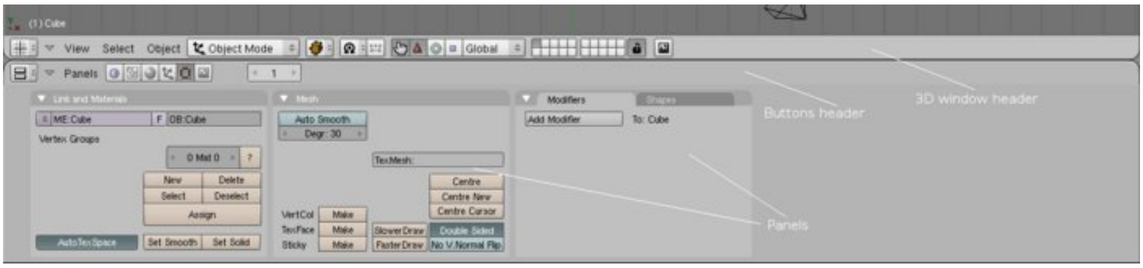
While you are using Blender be aware of what mode and types of object are selected. This helps in knowing what hotkeys work at what times.

Previous: Manual/PartI/Interface/Contexts	Contents	*Next:* Manual/PartI/Interface/Panels

User Manual: [Contents] | [Guidelines] | Blender Version **2.4x**

Panels

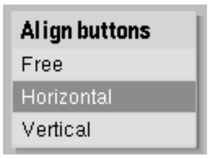
Panels generally appear in the Buttons window and by default the Buttons window is at the bottom; see (*Buttons window*). The Buttons window includes the Button window header and panels.



Buttons window

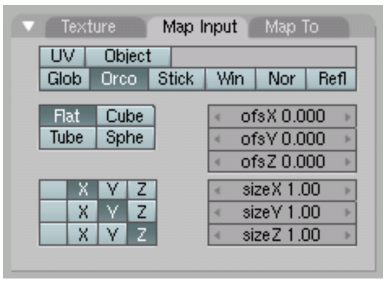
Each button on the Buttons header groups panels together into what is called a Context. And those Contexts are grouped further into Sub−Contexts. For example, all Material panels are grouped under the Shading context and Material sub−context.

The panels are not fixed in position relative to the window. They can be moved around the window by LMB clicking and dragging on the respective panel header.



Button Window Menu.

Panels can be aligned by RMB on the Buttons Window and choosing the desired layout from the Menu which appears (*Button Window Menu.*). Using MW scrolls the Panels in their aligned direction and CTRL MW and Ctrl MMB zooms the Panels in and out. Single Panels can be collapsed/expanded by LMB clicking the triangle on the left side of their header.



Panel with Tabs Example.

Particularly complex Panels are organized in *Tabs*. Clicking LMB on a Tab in the Panel header changes the buttons shown in (*Panel with Tabs Example.*). Tabs can be "torn out" of a Panel to form independent panels by clicking LMB on their header and dragging them out. In a similar way separate Panels can be turned into a single Panel with Tabs by dropping one Panel's header into another.

For further details about each panel see the Reference panels section.

Previous: Manual/PartI/Interface/Menus	Contents	*Next:* Manual/PartI/Interface/Buttons and Controls

User Manual: Contents | Guidelines | Blender Version **2.4x**

Buttons and Controls

Buttons are mostly grouped in the Button Window. But they can appear in other Windows.

Operation Button

An operation button

These are buttons that perform an operation when they are clicked (with LMB, as all buttons). They can be identified by their brownish color in the default Blender scheme (*An operation button*).

Toggle Button

Toggle buttons

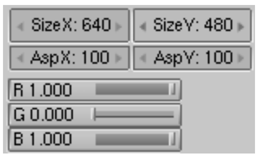
Toggle buttons come in various sizes and colors (*Toggle buttons*). The colors green, violet, and grey do not change functionality, they just help the eye to group the buttons and recognize the contents of the interface more quickly. Clicking this type of button does not perform any operation, but only toggles a state.

Some buttons also have a third state that is identified by the text turning yellow (the Emit button in *Toggle buttons*). Usually the third state means "negative," and the normal "on" state means "positive."

Radio Buttons

Radio buttons are particular groups of mutually exclusive Toggle buttons. No more than one Radio Button in a given group can be "on" at one time.

Number Buttons



Number buttons

Number buttons (*Number buttons*) can be identified by their captions, which contain a colon followed by a number. Number buttons are handled in several ways: To increase the value, click LMB on the right of the

button, where the small triangle is shown; to decrease it, click on the left of the button, where another triangle is shown.

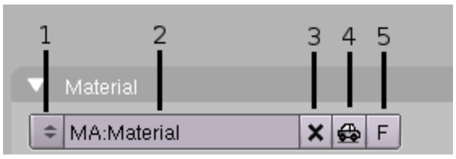
To change the value in a wider range, hold down LMB and drag the mouse to the left or right. If you hold CTRL while doing this, the value is changed in discrete steps; if you hold SHIFT, you'll have finer control over the values. ENTER can be used in place of LMB here.

You can enter a value directly by holding SHIFT and clicking LMB. You can also enter simple equations, like 3*2 instead of 6. Handy geometric constants to remember: pi is 3.14 and the square root of two is 1.414. Press SHIFT−BACKSPACE to clear the value; SHIFT−LEFTARROW to move the cursor to the beginning; and SHIFT−RIGHTARROW to move the cursor to the end. Press ESC to restore the original value.

Some number buttons contain a slider rather than just a number with side triangles. The same method of operation applies, except that single LMB clicks must be performed on the left or on the right of the slider, while clicking on the label or the number automatically enters keyboard input mode.

Menu Buttons

Use the Menu buttons to choose from dynamically created lists. Menu buttons are principally used to link DataBlocks to each other. (DataBlocks are structures like Meshes, Objects, Materials, Textures, and so on; by linking a Material to an Object, you assign it.)



Datablock link buttons

You can see an example for such a block of buttons in (*Datablock link buttons*).

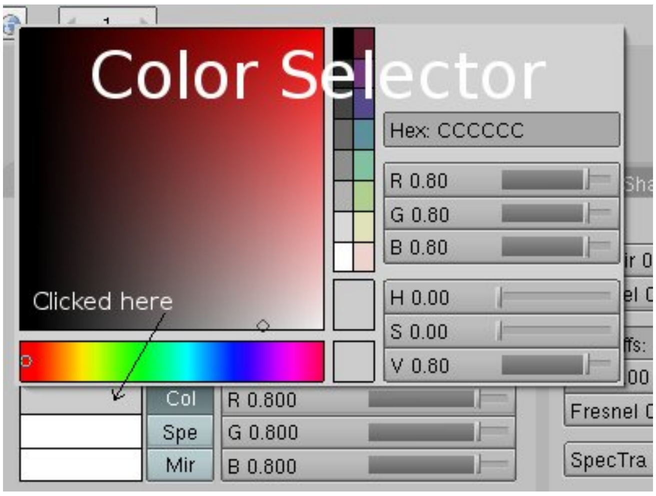
- 1 − The first button (with the tiny up and down pointing triangles) opens a menu that lets you select the DataBlock to link to by holding down LMB and releasing it over the requested item.
- 2 − The second button displays the type and name of the linked DataBlock and lets you edit its name after clicking LMB.
- 3 − The "X" button clears the link.
- 4 − The "car" button generates an automatic name for the DataBlock.
- 5 − And the "F" button specifies whether the DataBlock should be saved in the file even if it is unused (unlinked).

Unlinked objects

Unlinked data is ***not* lost until you quit Blender**. This is a powerful Undo feature. if you delete an object the material assigned to it becomes unlinked, but is still there! You just have to re−link it to another object or press the "F" button.

Color Selector controls

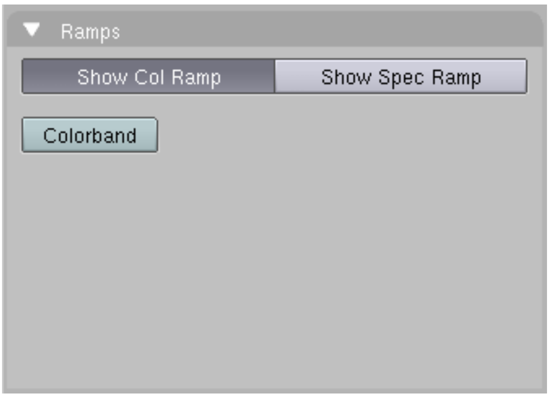
Some controls pop−up a dialog panel. For example, Color controls, when clicked, will pop up a Color Selector dialog; see (*Color Selector*).



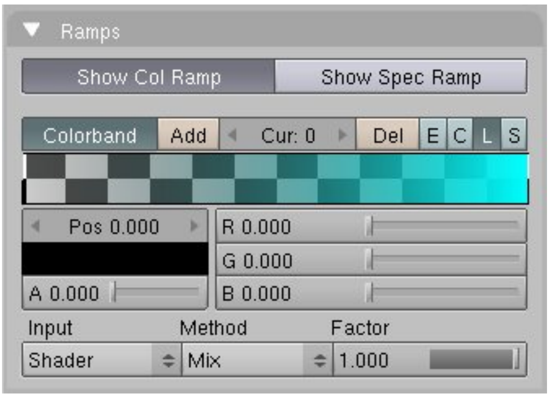
Color Selector

Cascade Buttons

Occasionally, some buttons actually reveal addition buttons. For example, the Ramps panel has a Cascade button called Colorband that reveals additional buttons dealing with colorbanding; see (*Colorband before*) and (*Colorband after*).



Colorband before



Colorband after

Previous: Manual/PartI/Interface/Panels	Contents	*Next:* Manual/Your_First_Animation_in_30_plus_30_Minutes_Part_I

 Your First Animation in 30 plus 30 Minutes Part I

Your First Animation in 30 plus 30 Minutes Part I

This chapter will guide you through the animation of a small "Gingerbread Man" character. We will describe each step completely, but we will assume that you have read the interface chapter, and that you understand the conventions used throughout this book.

In Part I of this tutorial we'll build a *still* Gingerbread Man. Then, in Part II, we will make him walk.

Note

For a much more in−depth introduction to Blender that focuses on character animation, check out the

Blender Summer of Documentation Introduction to Character Animation tutorial.

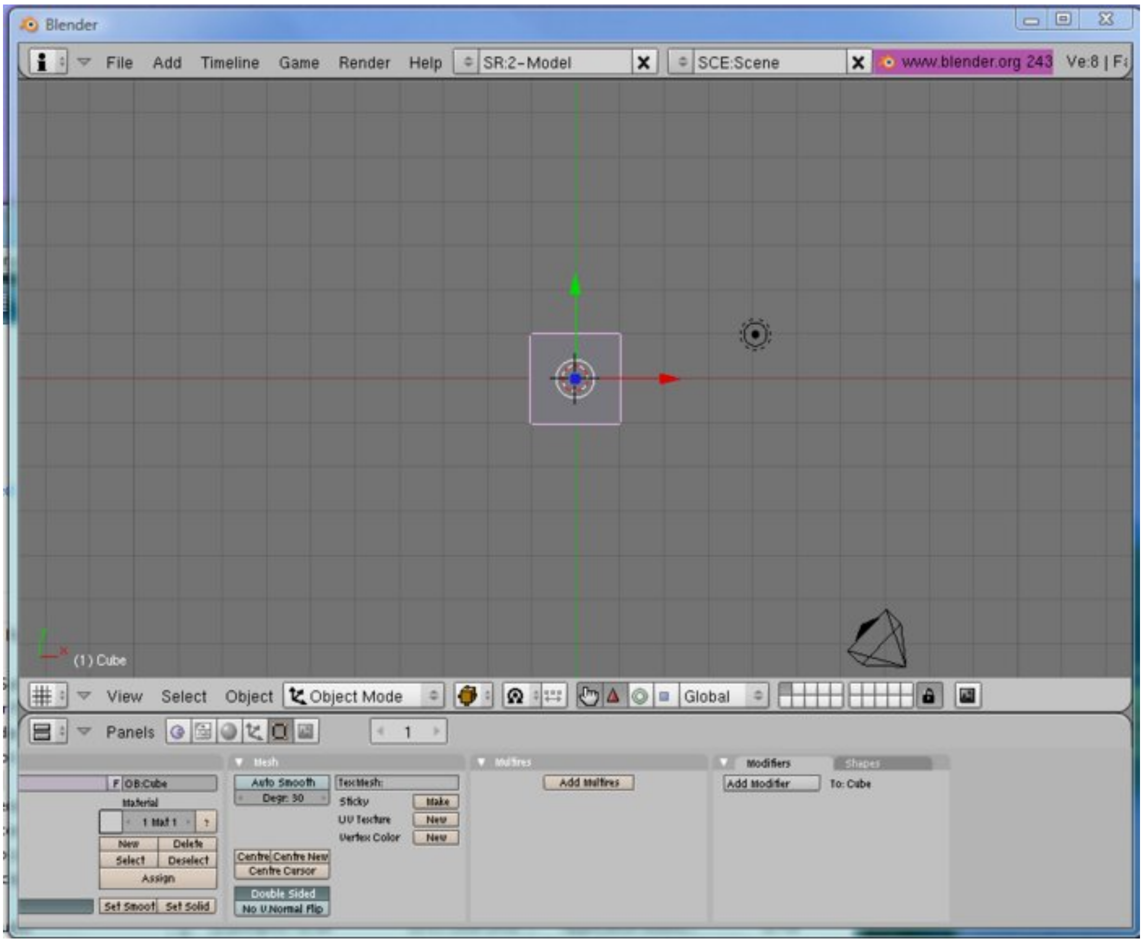
Just like the "Gus the Gingerbread Man" tutorial you see here, the BSoD Intro to Character Animation tutorial assumes no prior knowledge. It guides you through the process of making a walking, talking character from scratch and covers many powerful features of Blender not found here.

The BSoD Intro to Character Animation also has a downloadable PDF version (*3.75 MB*) for offline viewing.

Warming up

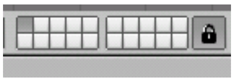
Let's start Blender. On the screen you should see, from the top view, the default set−up. A camera, a light, and a cube. The cube should already be selected, as indicated by its pink color.(*Default Blender screen as soon as you start it.*).

 Your First Animation in 30 plus 30 Minutes Part I



Default Blender screen as soon as you start it (small).

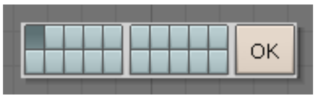
We will organize our working area by placing objects on different layers where we can hide them; we can also bring them back in the current scene whenever we need them. Here is how layers work:



Layer visibility controls.

Blender provides you with twenty layers to help to organize your work. You can see which layers are currently visible from the group of twenty buttons in the 3D window header (*Layer visibility controls*). You can change the visible layer with LMB and toggle visibility with Shift LMB. The last layer that is turned on becomes the active layer. The active layer is where all objects that will be created are stored.

So let's clean up the place.



Layer control toolbox.

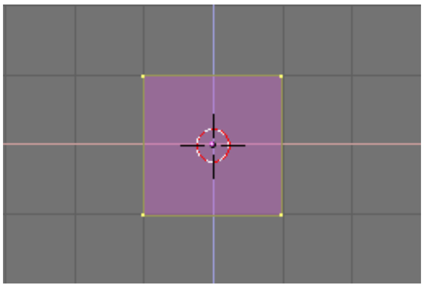
Select the camera and the lamp with Shift RMB and press M. A small toolbox, like the one in (*Layer control toolbox*), will appear beneath your mouse, with the first button checked, which means that the selected objects are stored in Layer1. Check the rightmost button on the top row and then click on OK. This will move

your camera and lamp to layer 10.

Now make sure that only Layer1 is visible, because we wouldn't want to erase the lamp or the camera ; select everything on that layer using A and erase it with X >> Erase Selected Object(s). This leaves all the room we can wish for to begin our modelling job.

Building the body

Change to the front view with NumPad 1 and add a cube −−if one is not present−− by pressing SPACE >> Add >> Mesh >> Cube. A cube will appear. Press TAB, and it will be in Edit Mode. (*Our cube in Edit Mode, all vertices selected*).



Our cube in Edit Mode, all vertices selected.

Edit Mode and Object Mode

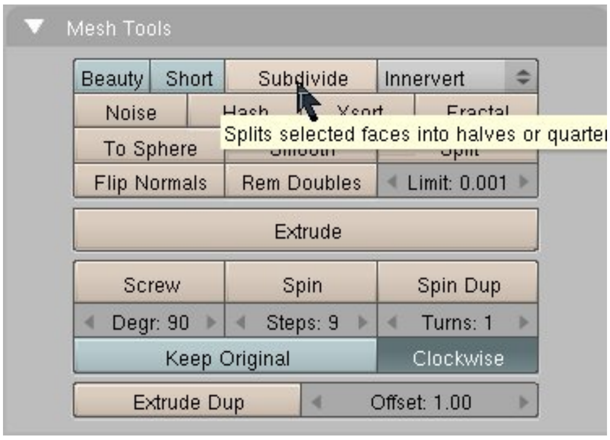
Edit Mode is a mode in which you can edit the vertices of the mesh. By default, all vertices are selected for every new object created (selected vertices are highlighted in yellow − unselected vertices are pink). In Object Mode, vertices cannot be selected or individually edited; the object can be changed only as a whole. You can press TAB to switch between these two modes, and the current mode is indicated in the header of the 3D window.

On many occasions you may have vertices hidden behind other vertices, as is the case here. Our subdivided cube has 26 vertices, yet you can only see nine because the others are hidden. A normal RMB click selects only one of these stacked vertices, whereas a box select selects them all. But beware that by default this is true only for the wireframe drawtype : in any other mode, Shaded, Solid or Textured we can only select visible vertices, even with box select. To select vertices that are hidden behind others uncheck the limit selection button (Limit selection button).}}

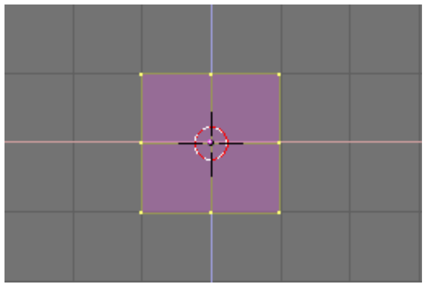
We will call our Gingerbread man "Gus". Our first task is to build Gus's body by working on the vertices of our Cube. To see the Blender tools that we'll use for this purpose, press the button showing a square with yellow vertices in the Button window header (*The Edit Buttons Window button*), or press F9

The Edit Buttons Window button.

Now locate the Subdivide button in the Mesh Tools panel and press it once (*The Mesh Tools panel in the Edit context (F9)*). This will split each side of the cube in two, creating new vertices and faces (*The cube, subdivided once*).



The Mesh Tools panel in the Edit context (F9).



The cube, subdivided once.

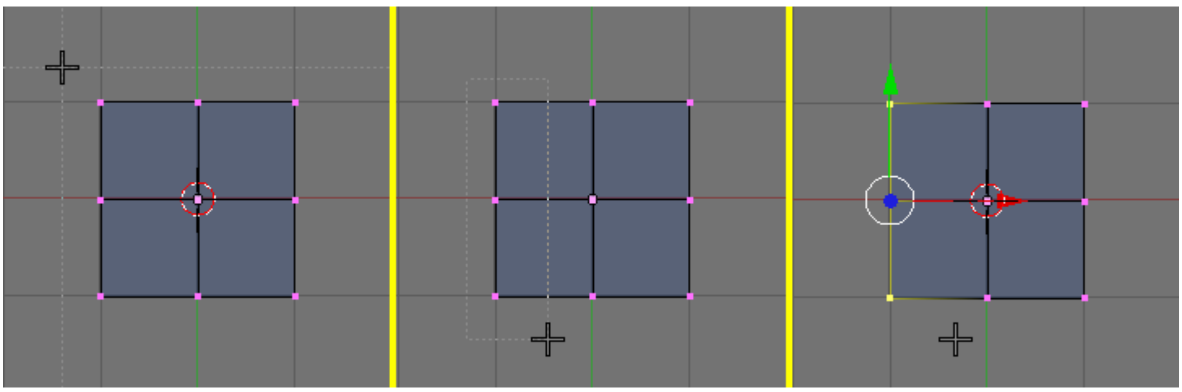
With your cursor hovering in the 3D window press A to deselect all elements. Vertices will turn pink.

Limit selection button

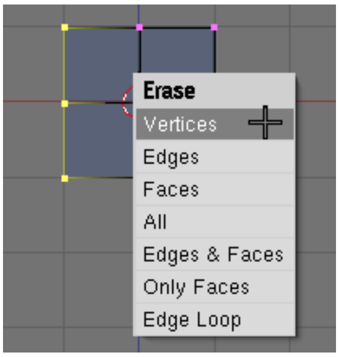
Box Select

On many occasions you may have vertices hidden behind other vertices, as is the case here. Our subdivided cube has 26 vertices, yet you can only see nine because the others are hidden. A normal **RMB** click selects only one of these stacked vertices, whereas a box select selects them all. But beware that by default this is true only for the wireframe drawtype : in any other mode, Shaded, Solid or Textured we can only select visible vertices, even with box select. To select vertices that are hidden behind others uncheck the **Limit selection button**. The button is selected in the image at the right.
You must have the **Limit selection button** *unselected*  to continue this tutorial.

Now press B, the cursor will change to a couple of orthogonal grey lines. Move the cursor above the top left corner of the cube, press and hold LMB , then drag the mouse down and to the right so that the grey box encompasses all the leftmost vertices. Now release the LMB  (*The sequence of Box selecting a group of vertices*).



The sequence of Box selecting a group of vertices.

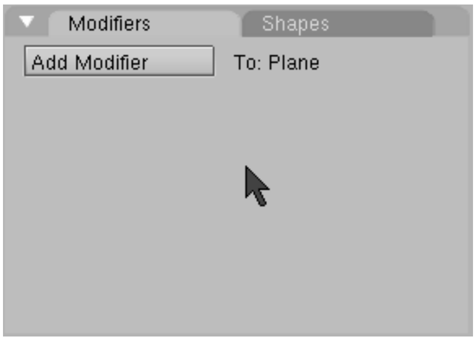


The pop−up menu of the Delete (X) action.

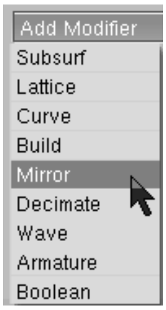
Press X and from the popup menu select Vertices to erase the selected vertices (*The pop−up menu of the Delete (X) action*).

Mirror modelling

To model symmetrical Objects we can use the Mirror modifier. It allows us to model only one side of Gus while Blender creates the other in real time. Go to the Edit context (F9) and find the Modifiers panel, (*The modifiers panel*).



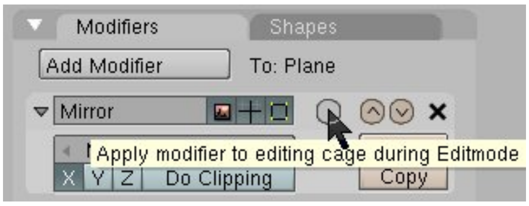
The modifiers panel.



List of modifiers.

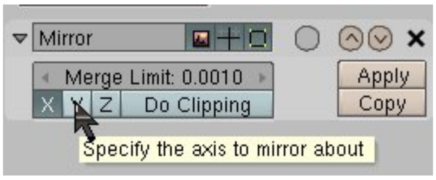
It is pretty empty for the moment. Clicking the button marked *Add Modifier* opens a list from which you'll choose Mirror(*List of modifiers*).

Nothing much seems to happen; that is because the modifiers offer quite a bit of control over what's displayed and what's not. In our case we will check the *Cage Mode* button so we can see the transparent faces in Edit Mode, (*Cage Mode button*).



Cage Mode button.

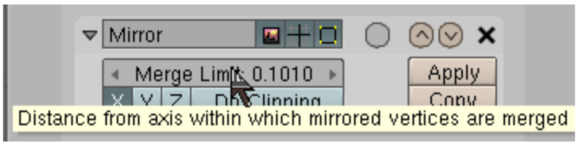
We choose the axis that will run from the modelled side of our character to the side Blender is completing by checking either the X, Y or Z button; the mirror plane is perpendicular to that axis. In our case it is the X−axis, (*Axis perpendicular to the mirror plane*).



Axis perpendicular to the mirror plane.

The Merge Limits button (*Merge Limits button*) acts as a safety net. Any vertex closer to the mirror plane, than the limit we set, will be placed exactly on the mirror plane. The limit can be set from **0.000** to **1.000** units and how big it should be depends on the nature and the scale of the current job.

For modelling Gus, a vertex that would be more than **0.1** units away from the mirror plane would be noticeable but anything closer might not. Our mesh could end up ripped in the middle if vertices that should be on the mirror plane aren't. To avoid inadvertently neglecting a wandering vertex, we should set the Merge Limits to **0.1**.



Merge Limits button.

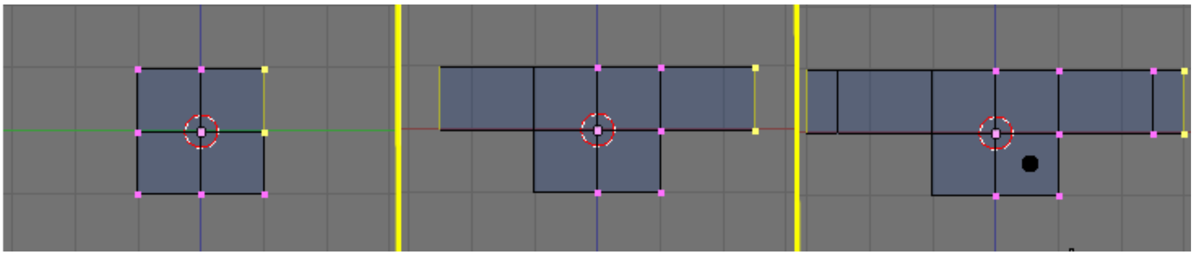
Finally, with the *Do Clipping* button checked (*Do Clipping button*), our mirror becomes a frontier that no vertex can cross. If this were to happen it would cause quite a mess. Also, when *Do Clipping* is active, every vertex that is **on** the mirror sticks to it.

Do Clipping button.

As you can see, the Mirror modifier gives us a lot of features to make our lives easier.

Arms and Legs

Let's create Gus's arms and legs. Using the sequence you just learned, Box Select the two top−right−most vertices (*Extruding the arm in two steps*), left) which will actually select the other two behind them, for a total of four vertices. Press E and click on the Region menu entry to extrude them. This will create new movable vertices and faces which you can move with the mouse. Move them one and a half squares to the right, then click LMB to fix their position. Extrude again with E then move the new vertices another half a square to the right. (*Extruding the arm in two steps*) shows this sequence.



Extruding the arm in two steps.

Undo/Redo

Blender has two Undo features, one for Edit Mode and the other for Object Mode.

In Edit Mode press Ctrl Z to Undo and keep pressing Ctrl Z to roll back changes as long as the Undo buffer will allow; Shift Ctrl Z re−does changes. Alt U opens a menu with a list of possible undos so that you can easily find the point you want to revert to.

Two things to remember:

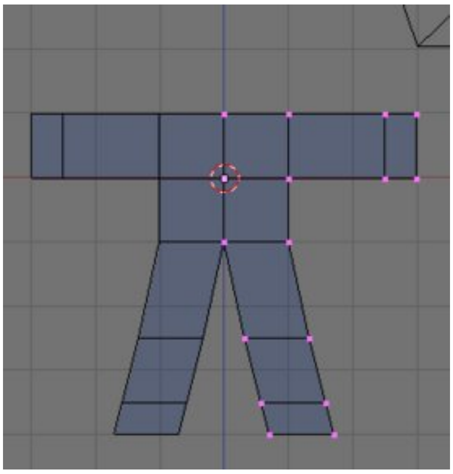
- Undo in Edit Mode works only for the Object currently in that mode.
- Undo data is not lost when you switch out of Edit Mode, but it is as soon as you start editing a *different* Object in Edit Mode.

In Object Mode the same shortcuts apply. Ctrl Z to undo, Shift Ctrl Z to redo and Alt U to see the history. If you made changes in Edit Mode that are not lost for that Object, they will all be undone in one single shot with Ctrl Z when this step −−marked as Edit Mode in the Object Mode (Alt U) history−− has its turn.

If you change your mind in the middle of an action, you can cancel it immediately and revert to the previous state by pressing ESC or RMB .

Coincident vertices

Extruding works by first creating new vertices and then moving them. If in the process of moving you change your mind and press ESC or RMB to cancel, the new vertices will still be there, on top of the original ones! The simplest way to go back to the state before you started extruding is to Undo (Ctrl Z). It is sometimes useful to intentionally create new vertices this way and then move, scale or rotate them by pressing G,S or R.



Body.

Gus should now have a left arm that you modelled (he's facing us) and a right arm that Blender added. We will build the left leg the same way by extruding the lower vertices three times. Try to produce something like in (*Body*). If you are using *Extrude − Region*, you will have to clear the transformation constraint (by clicking with MMB) if you want to move the new vertices around freely. Otherwise your legs will end up going straight down, rather than down and to the side as in (*Body*).

Note

You will need to uncheck "Do Clipping" if you were using it before (otherwise Gus will end up with a skirt rather than pants).

We're done with mirror modelling. In the next steps we will experiment with other techniques. We need to make the right part of our model *real* since nothing done with modifiers is permanent unless we *apply* the changes. With Gus being in Object Mode (press TAB), click on the *Apply* button of the Mirror modifier.

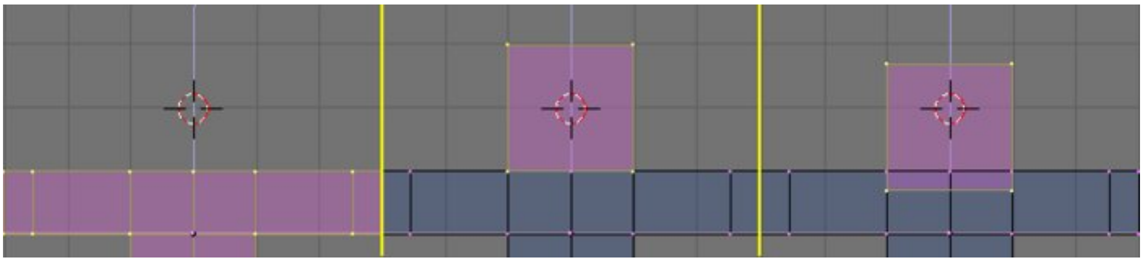
The Head

Gus needs a head.

Change back to Edit Mode (press TAB)

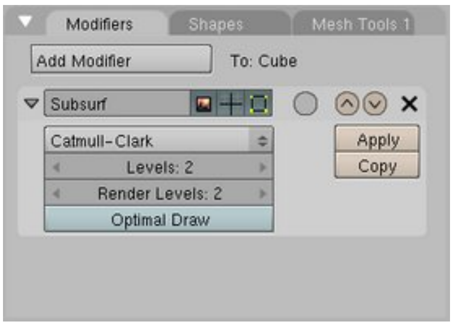
Move the cursor to exactly one square above Gus's body (leftmost image of *Adding the head*) and add a new cube (SPACE>ADD>Cube). To place the cursor at a specific grid point, position it next to where you want it and press **SHIFT+S** to bring up the Snap Menu. **Cursor to Grid** places the cursor exactly on a grid point. That's what we want right now. **Cursor to Selection** places it exactly on the selected object, which is sometimes handy.

Now press G to switch to Grab Mode and move the newly created vertices down, constraining the movement by moving the head down a bit and clicking MMB, for about one third of a grid unit (rightmost image of *Adding the head.*).



Adding the head.

SubSurfaces *(subsurf)*



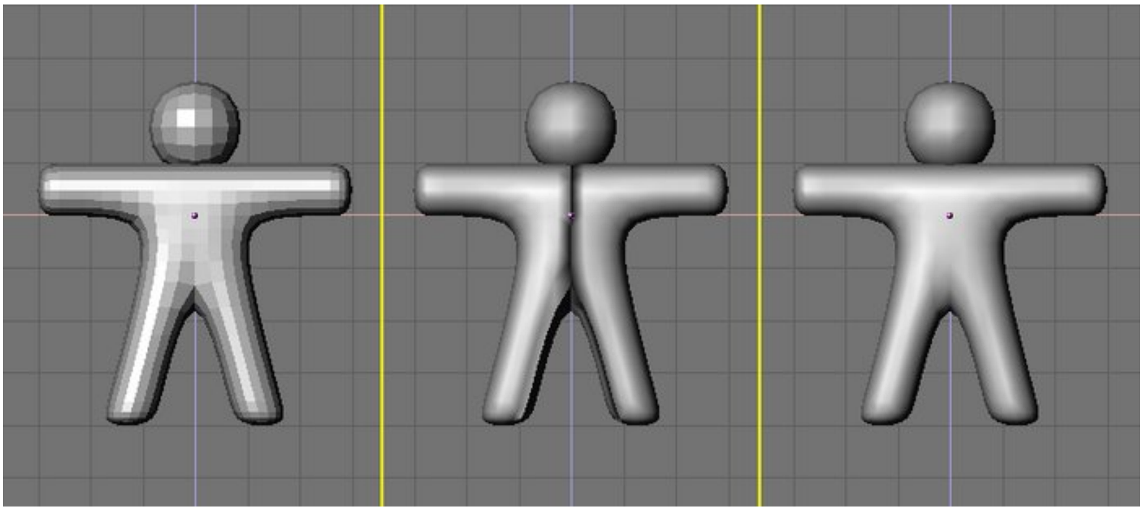
The *Subsurf* modifier in the Modifiers panel of the Editing context (F9)

So far what we have produced is a rough figure at best. To make it smoother, locate the Modifier panel in the *Editing* context (F9) and add a Subsurf modifier, (*The* Subsurf *modifier in the Modifiers panel*). Be sure to set both *Levels* NumButtons below or at 2. The first Level is for what you'll see in the 3D Window area, the second for the renderer.

SubSurfaces

SubSurfacing is an advanced modelling tool, it dynamically refines a given coarse mesh creating a much denser mesh and locating the vertices of the finer mesh so that they smoothly follow the original coarse mesh. The shape of the Object is still controlled by the location of the coarse mesh vertices, but the rendered shape is a finely smooth mesh.

- Switch out of Edit Mode (TAB) and from the current Wireframe mode to Solid mode using Z to have a look at Gus. He should look like (*Setting Gus to smooth*, left).



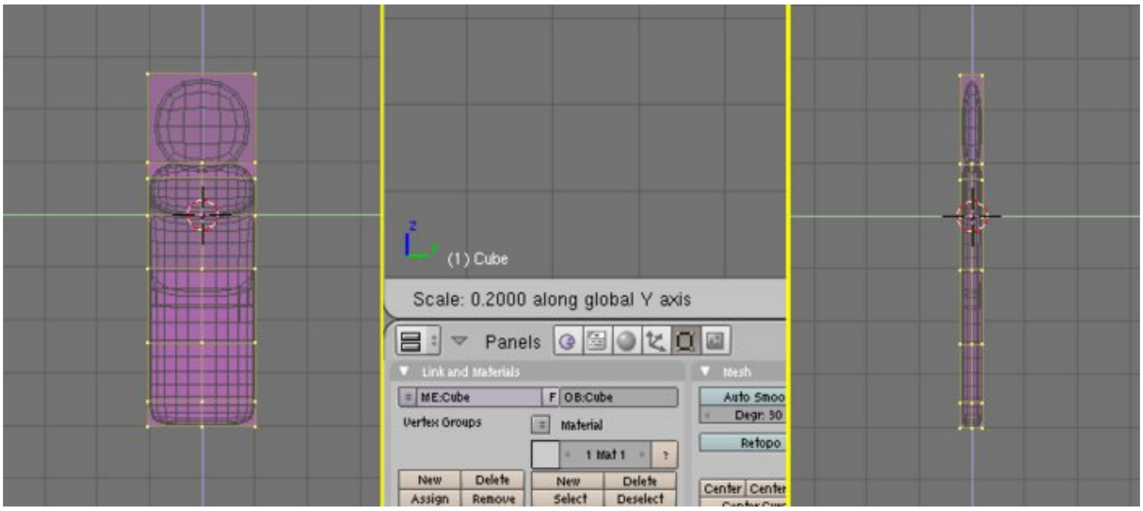
Setting Gus to smooth.

- To make Gus look smooth, press the SetSmooth button found in the Link and Material panel of the Editing context (F9). Gus will now appear smooth although he may wear some funny black lines in his middle. This is usually avoided if you used the Mirror Modifier but it might happen when extruding and flipping, as it was done before the modifier was introduced. (*Setting Gus to smooth.*, middle). These lines appear because the SubSurf's finer mesh is computed using information about the coarse mesh normal directions, which may not all point in the right direction, that is, some face normals might point outward and some inward. To reset the normals, switch back to Edit Mode (TAB), select all vertices (A), and press Ctrl N. Click with LMB on the Recalculate normals outside box which appears. Now Gus should be nice and smooth (*Setting Gus to smooth*, right).

Press MMB and drag the mouse around to view Gus from all angles. Oops, he is too thick!

Constrained Scaling

(*Slimming Gus using constrained scaling.*, left).



Slimming Gus using constrained scaling.

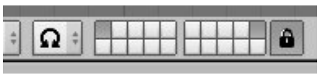
Let's make Gus thinner:

- Switch to Edit Mode if you are not there already (TAB), then back to Wireframe mode (Z), and select all vertices with A. You can do the following steps just as well in Object Mode, if you like.
- Press S and start to move the mouse horizontally. (Click MMB to constrain scaling to just one axis or press Y to obtain the same result). If you now move the mouse toward Gus he should become thinner but remain the same height.
- The header of the 3DWindow toolbar shows the scaling factor (*Slimming Gus using constrained scaling.*, center). Press and hold CTRL. The scale factor will now vary in discrete steps of value 0.1. Scale Gus down so that the factor is 0.2, then set this dimension by clicking LMB, (*Slimming Gus using constrained scaling.*, right).
- Return to Front view and to Solid mode (Z), then rotate your view via MMB. Gus is much better now!

Let's see what Gus looks like

We're just about ready to see our first rendering, but first, we have some work to do.

- Switch to Object Mode if not already there (TAB).



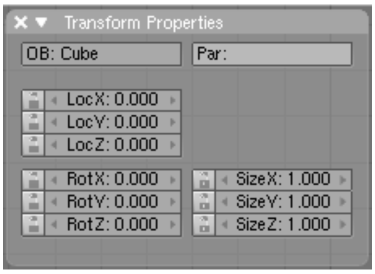
Making both layer 1 and 10 visible.

- Shift LMB on the top right small button of the layer visibility buttons in the 3DWindow toolbar (*Making both layer 1 and 10 visible.*) to make both Layer 1 (Gus's layer) and Layer 10 (the layer with the camera and the lamp) visible.

A Tip

Remember that the last layer selected is the active layer, so all subsequent additions will automatically be on layer 10.

- Press N to bring up the Transform Properties window (*The Panel for numerical input of object position/rotation etc*). The location of the camera is specified by LocX, LocY, and LocZ.



The Panel for numerical input of object position/rotation etc.

- Select the camera (RMB) and move it to a location like (x=7, y=−10, z=7). Do this by pressing G and dragging the camera. You may need to change views and move the camera a second time to adjust all three coordinates. If you prefer to enter numerical values for an object's location you can do so by holding SHIFT and clicking LMB on a NumButton and then entering the desired value. Remember to press ENTER to confirm your input.

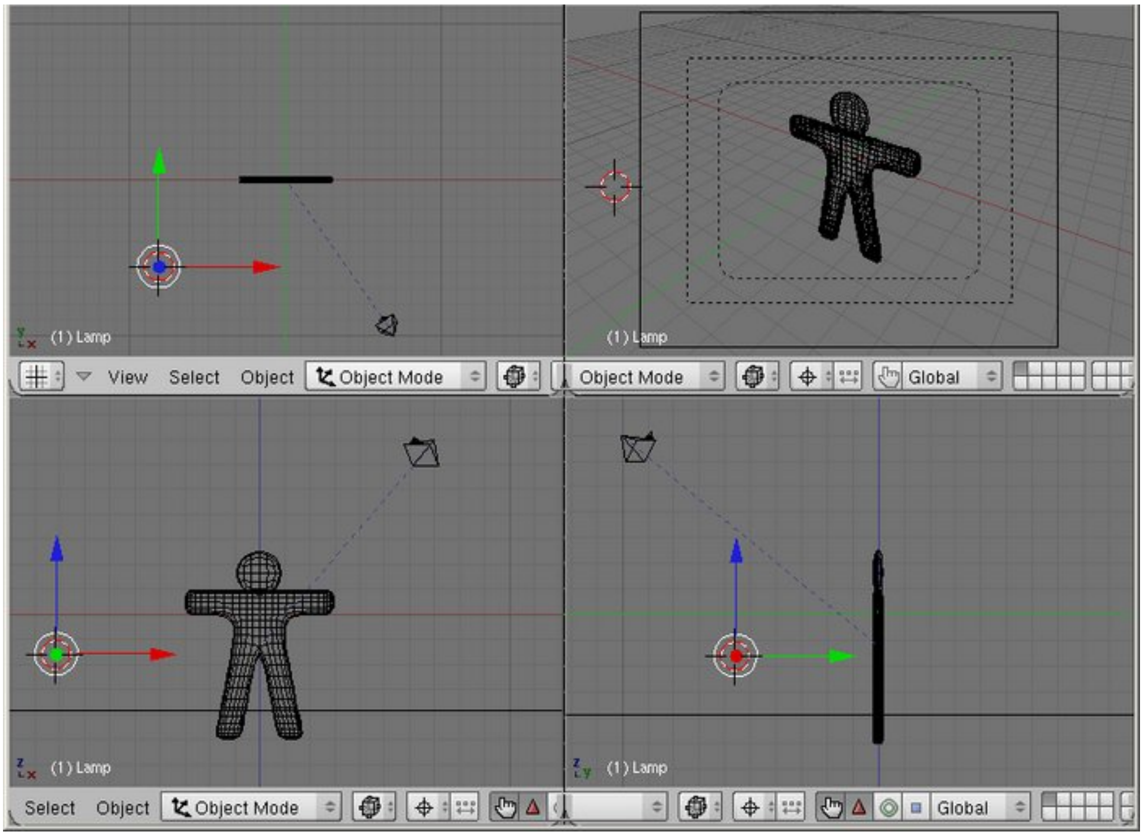
Camera setup

To make the camera point at Gus, keep your camera selected then select Gus via Shift RMB . The camera should be magenta and Gus light pink. Now press Ctrl T and select the TrackTo Constraint entry in the pop up. This will force the camera to track Gus and always point at him. This means that you can move the camera wherever you want and be sure that Gus will always be in the center of the camera's view.

Tracking

If you choose the option **Old Track** and the camera has a rotation of its own, as is often the case, it could point in an unexpected direction. In that case select the tracking object (in our example the camera), and press **ALT−R** to remove the object's rotation. Once you do this the camera will really track Gus.

(*Camera position with respect to Gus*) shows top, front, side and camera view of Gus. To obtain a camera view press NumPad 0 or select View>>Camera.



Camera position with respect to Gus.

The Ground

Now we need to create the ground for Gus to stand on.

- In top view (NumPad 7 or View>>Top), and in Object Mode, add a plane

(SPACE>>Add>>Mesh>>Plane).

Note

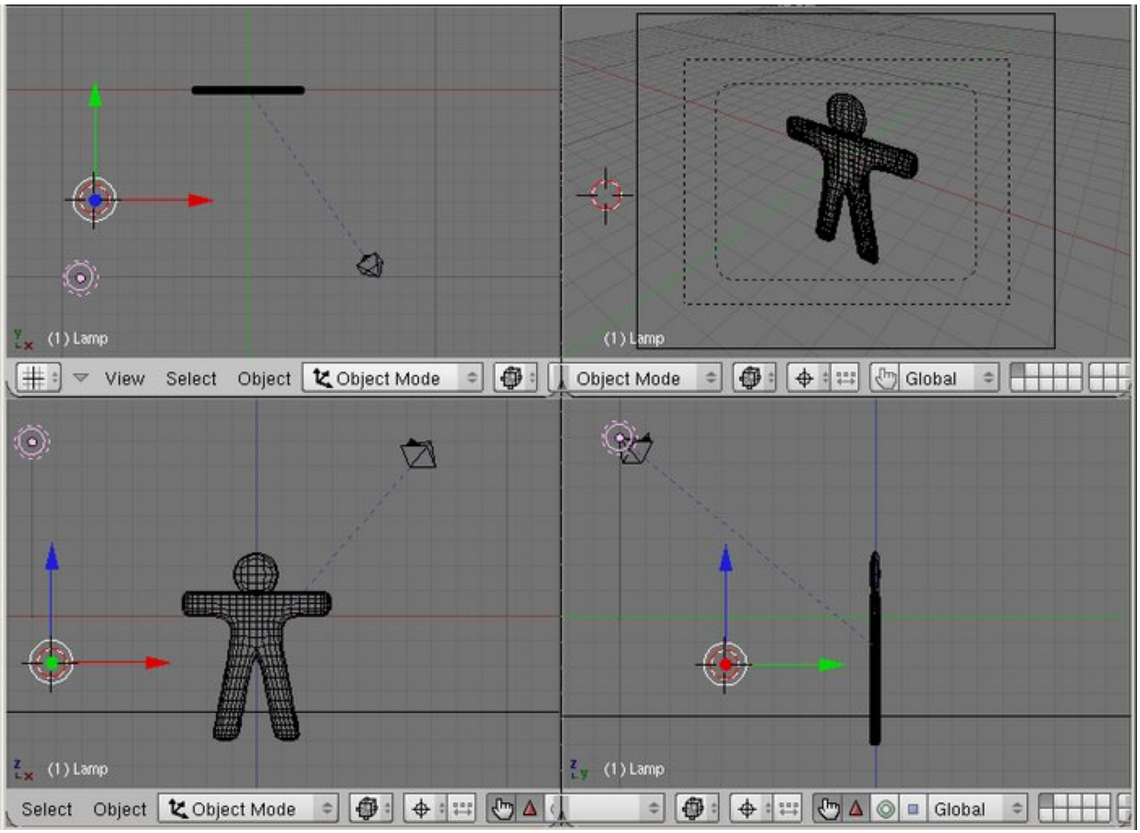
It is important to be out of Edit Mode, otherwise the newly added object would be part of the object currently in Edit Mode, as when we added Gus' head.

- Switch to the Front view (NumPad 1 or View>>Front) and move (G) the plane down to Gus's feet, using CTRL to keep it aligned with Gus.
- Go to Camera view (NumPad 0 or View>>Camera) and, with the plane still selected, press S to start scaling.
- Enlarge the plane so that its edges extend beyond the camera viewing area, as indicated by the outer white dashed rectangle in Camera view.

Lights

Now, lets add some light!

- In Top view (NumPad 7), move the existing Lamp light (if you do not have a Lamp light in your scene you can add one with SPACE>>Add>>Lamp>>Lamp) in front of Gus, but on the other side of the camera; for example to (x= −9, y= −10, z=7) (*Inserting a Lamp.*).



Inserting a Lamp.

The Lamp buttons window button.

- Switch to the Shading context (F5) and then the Lamp buttons window via the sub−context button with a lamp in the Button Window header (*The Lamp buttons window button.*).

- In the Buttons Window, Preview Panel, press the Spot toggle button to make the lamp a Spotlight (*Spot light settings.*) of pale yellow (R=1, G=1, B=0.9). Adjust Samples: to 4 and SpotBl: to 1.0.

Spot light settings.

- Make this spotlight track Gus just as you did for the camera by selecting Spot, SHIFT, then Gus, then by pressing Ctrl T>>TrackTo Constraint.
- Add a second lamp that provides more uniform fill light via (SPACE>>Add>>Lamp>>Hemi). Set its Energy to 0.5 (*Hemi lamp settings*). Move it a little above the camera (x= 7, y= −10, z=9) and set it to track Gus as before.

Hemi lamp settings

Two lamps?

Use two or more lamps to help produce soft, realistic lighting, because in reality natural light never comes from a single point.

Rendering

We're almost ready to render. As a first step, press the Scene context button and the Render sub−context button in the Button window header (*The Rendering buttons window buttons.*).

The Rendering buttons window buttons.

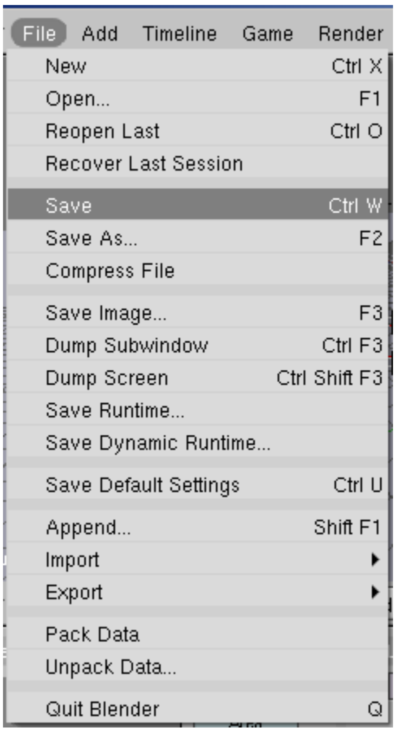
We will use the default rendering settings, as shown in (*The Rendering Buttons window*).

The Rendering Buttons window

Now press the RENDER button or F12. The result, shown in (*Your first rendering. Congratulations!*), is actually quite poor. We still need materials, and lots of details, such as eyes, and so on.

Your first rendering. Congratulations!

Saving our work



The Save menu.

If you have not done so already, now would be a good time to save your work, via the File>>Save menu shown in *The Save menu.*, or Ctrl W. Blender will warn you if you try to overwrite an existing file.

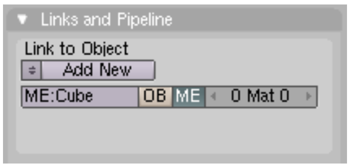
Blender does automatic saves into your system's temporary directory. By default, this happens every four minutes and the file name is a number. Loading these saves is another way to undo unwanted changes.

Materials and Textures

It's time to give Gus some nice cookie−like material.

The Material Buttons window Button.

- Select Gus. Then, in the Button Window header, select the Shading Context by pressing the red dot button (*The Material Buttons window Button.*) or pressing F5. Then press the red dot sub−context button to access the Material panels.

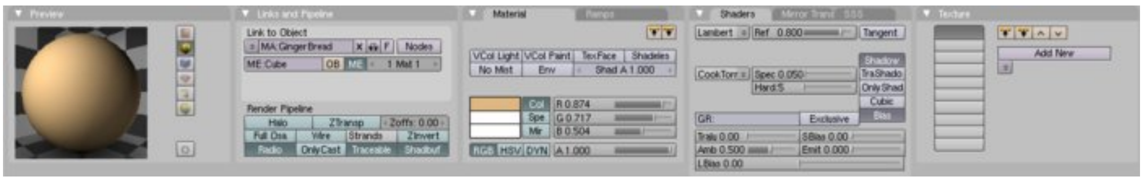


The Material Menu button.

- The Button window will be almost empty because Gus has no materials yet. To add a material, click on the Menu Button in the Material Panel (the one with two triangles, pointing up and down) and select Add New (*The Material Menu button.*).

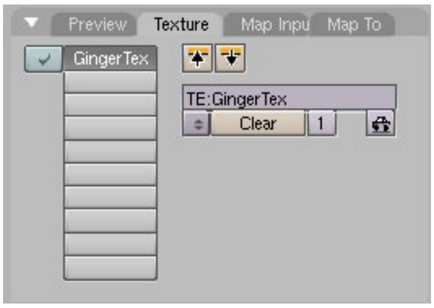
- The Buttons window will be populated by Panels and Buttons and a string holding the Material name, something like "Material.001", will appear next to the white square button. Click the name and change it to something meaningful, like "GingerBread" (don't type the quotes).

- Modify the default values as per (*The Material Buttons window and a first gingerbread material*) to obtain a first rough material. Note that you must click the Shaders tab to reveal the shader panel.



The Material Buttons window and a first gingerbread material.

- Press the Menu Button in the Textures Panel area (*The Textures menu button in the Material Buttons*) and select Add new. We're adding a texture in the first channel. Call it "GingerTex."

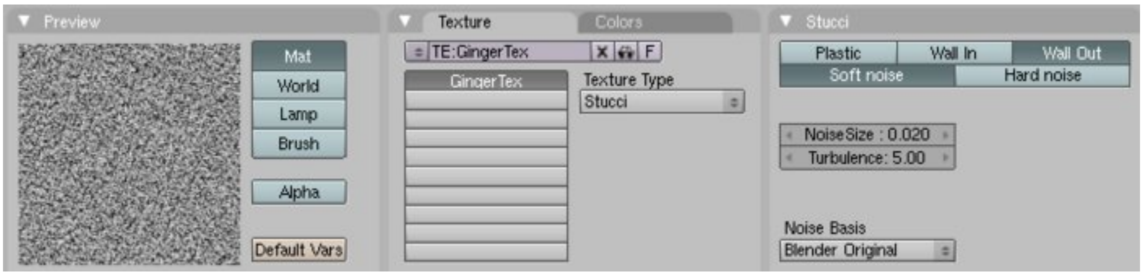


The Textures menu button in the Material Buttons

The Texture Buttons window Button.

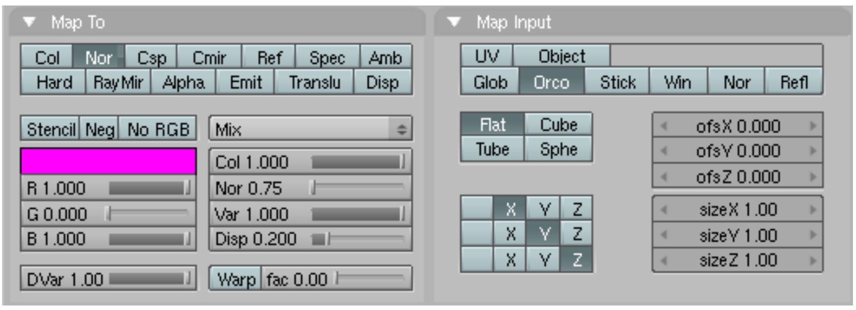
- Select the Texture Buttons by clicking the button in (*The Texture Buttons window Button*) or by pressing F6.

- From the columns of ToggleButtons which appear in the Texture panel select Stucci and set all parameters as in (*The Texture Buttons window with a stucci texture*).



The Texture Buttons window with a stucci texture.

- Return to the Material buttons (F5) and set the Map Input and Map To tabs of the Texture Panel as in (*Settings for the Stucci texture in the Material Buttons window*). Release the Col Toggle Button and set the Nor Toggle Button, then raise the Nor slider to **0.75**. These changes will make our Stucci texture act as a "bumpmap" and make Gus look more biscuit−like.



Settings for the Stucci texture in the Material Buttons window.

- Now add a second texture, name it "Grain", and make it affect only the Ref property with a **0.4** Var (*Settings for an additional Noise texture in channel 2*). The texture itself is a plain Noise texture.

Settings for an additional Noise texture in channel 2.

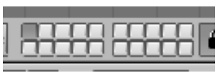
- Give the ground an appropriate material, such as the dark blue one shown in (*A very simple material for the ground*).

A very simple material for the ground.

Eyes and detail

To give some finishing touches we'll add eyes and some other details.

- First make Layer 1 the only one visible by clicking with LMB on the layer 1 button (*Layer visibility buttons on toolbar*). This will hide the lamps, camera, and ground.

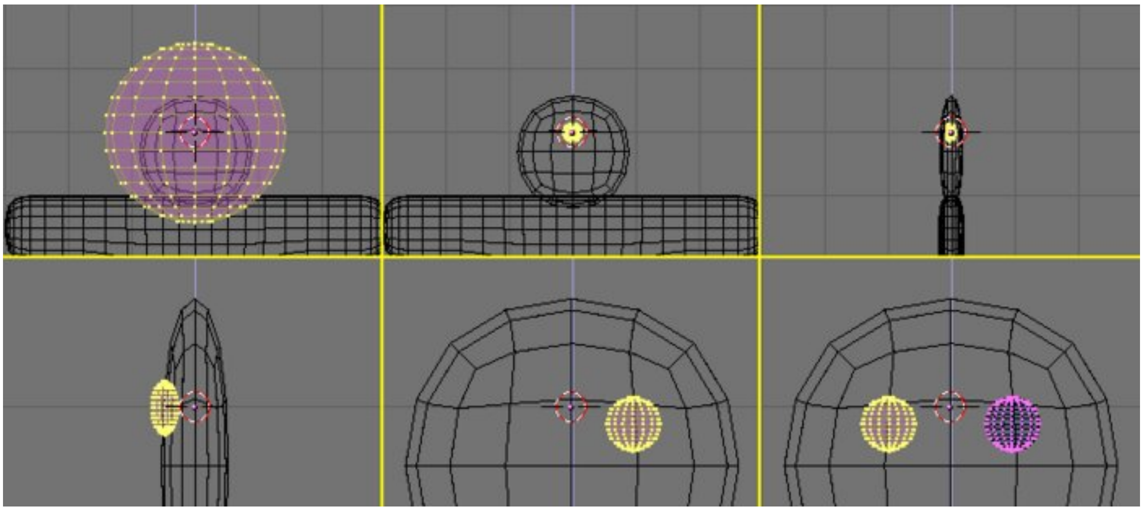


Layer visibility buttons on toolbar.

- Place the cursor at the center of Gus's head. (Remember that you are in 3D so be sure to check at least two views to be sure!)

- In Object Mode, add a sphere (SPACE>>ADD>>Mesh>>UVsphere). You will be asked for the number of Segments: (meridians) and Rings: (parallels) into which to divide the sphere. The default of **32** is more than we need here, so use a value of **16** for both. The sphere is in the first image at the top left of the sequence in (*Sequence for creation of the eyes*).

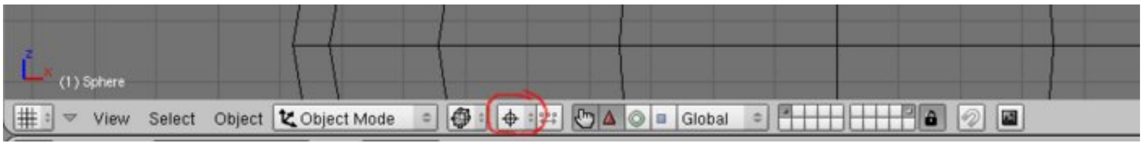
- Scale the sphere down (S) to a factor of about **0.15** in all dimensions, then switch to side view (NumPad 3) and scale it only in the horizontal direction (Y) a further **0.5**, see the second two images in (*Sequence for creation of the eyes*).



Sequence for creation of the eyes.

- Zoom a little if necessary via NumPad +, MW, or Ctrl MMB, and drag the sphere (G) to the left so that it is halfway into the head, as shown in the first image in the second row of (*Sequence for creation of the eyes*).

- Return to front view (NumPad 1) and move the sphere sideways, to the right. Place it where Gus should have an eye.



The crosshair pivot button.

- Flip a duplicate around the cursor:

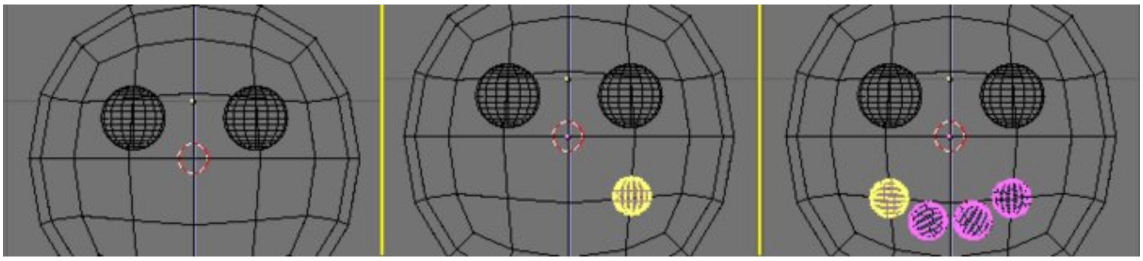
  Select the crosshair pivot button in the header of the 3D window (*The crosshair pivot button*). In Edit Mode, press A to select all, Shift D to duplicate, and ESC to stop placing it with the mouse. Then press M to mirror followed by X to mirror around the X axis. Return the pivot button to its default setting (Median Point).

Mirroring

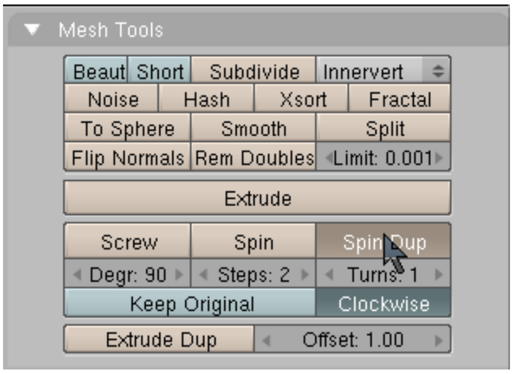
Mirroring can also be done in object mode using CTRL M.
Now Gus has two eyes.

Mouth

- Exit Edit Mode (TAB), and place the cursor as close as you can (remember the Shift S key) to the center of Gus's face. Add a new sphere and scale and move it exactly as before, but make it smaller and place it lower than and to the right of the cursor, centered on the SubSurfed mesh vertex *Creating a mouth with Spinning tools.*).



Creating a mouth with Spinning tools.



The Spin Tools buttons in the Edit Buttons window.

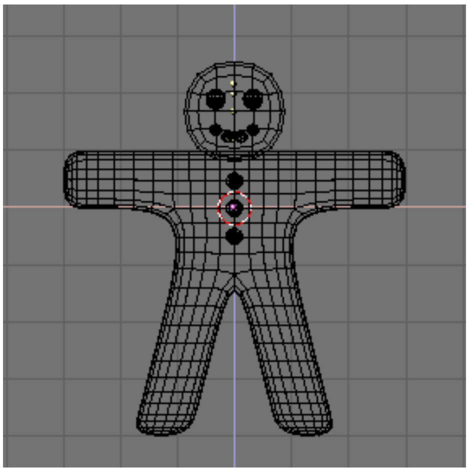
- Switch to Edit Mode (TAB). Now, in the Edit Buttons (F9), locate the group of buttons at bottom in the Mesh Tools Panel (*The Spin Tools buttons in the Edit Buttons window.*). Set Degr: to **90**, Steps: to **3**, and verify that the Clockwise: TogButton is on. Then, with all vertices still selected, press SpinDup. This will create three duplicates of the selected vertices on an arc of **90** degrees, centered around the cursor. The result should be Gus's mouth, like the last image of the sequence shown in *Creating a mouth with Spinning tools.*.

- Now go back to Object Mode and add three more spheres to form Gus's buttons. Once you have made one button, you can simply exit Edit Mode, press Shift D to create a duplicate, and move the

duplicate into place, as shown in *The complete Gus!*.

Attaching the spheres

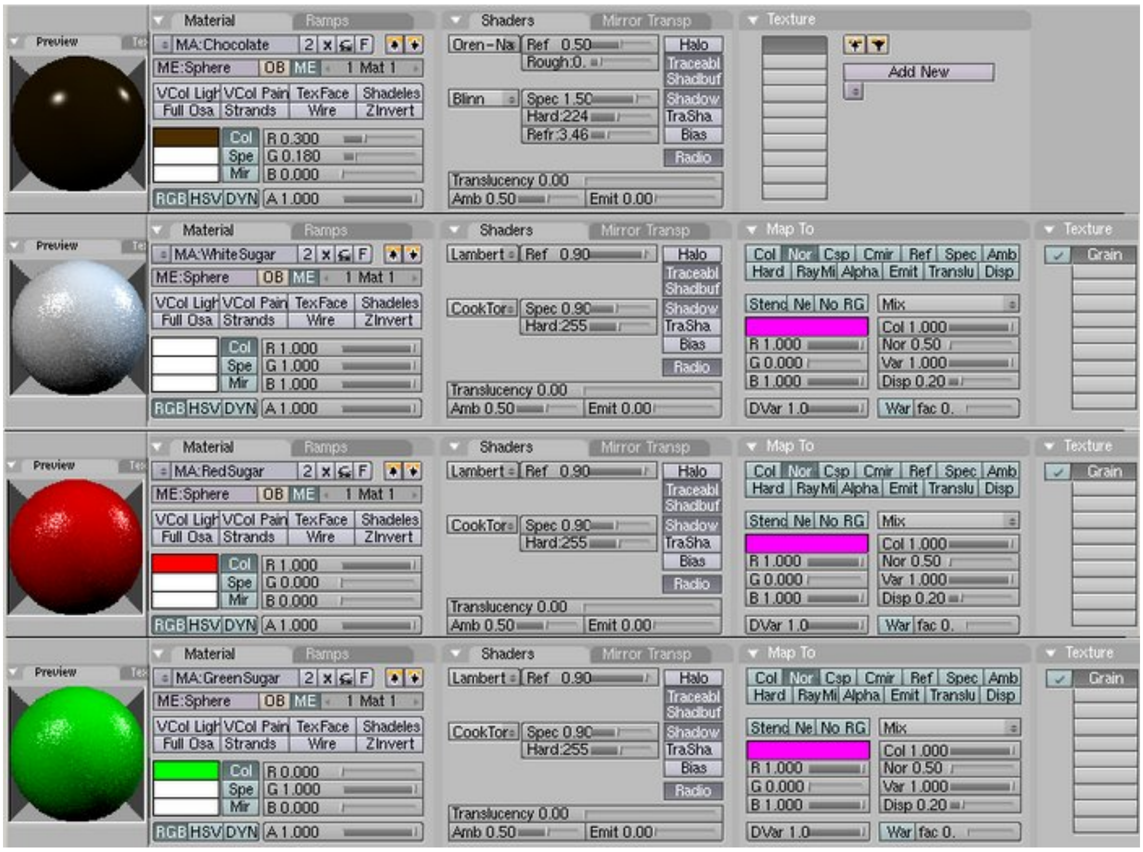
If we want to be able to grab Gus and move him around as a whole (this goes beyond the animation in the second part of this tutorial), we now need to attach the small spheres representing eyes, mouth, and buttons to the body. Enter Object Mode. Press A until nothing is selected. Now right click one sphere (if more than one is selected as a group, that's ok). Holding SHIFT, select the body. Then hit CTRLP and left click Make parent on the pop up. Deselect everything and repeat to attach each element.



The complete Gus!

Eyes material

Give the eyes a chocolate−like material, like the one shown at the top in *Some other candy materials.*. Give the mouth a white sugar like material, like the second one shown in (*Some other candy materials*), and give the buttons a red, white, and green sugar like material. These are shown from top to bottom in (*Some other candy materials*) too.



Some other candy materials.

Material Menu.

Objects sharing a material

To give one object the same material as another object, select that material in the **Material Menu** list which appears when you press the Menu Button ButtonWindow **Material** Panel, see (*Material Menu*).

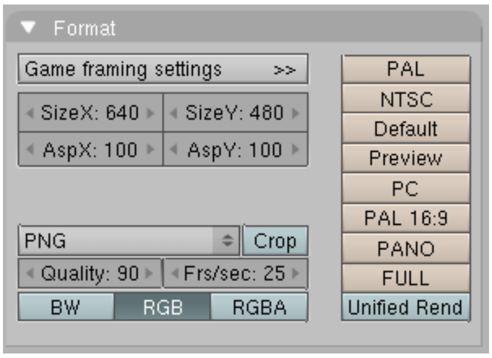
Rendering

Once you have finished assigning materials, make layer **10** visible again (remember how? Hint, look at the 3D window header), so that lights and the camera also appear, and do a new rendering (F12).

The result should look more or less like (*The complete Gus still rendering*).

The complete Gus still rendering.

Saving



File type selection menu in the Rendering Buttons window.

Save your image by pressing F3. Enter the name of your image in the file window and save.

You must choose the image format (JPEG, PNG, and so on) by setting it in the Rendering buttons *before* pressing F3 (*The Rendering buttons window buttons*) and using the Menu (*File type selection menu in the Rendering Buttons window*) in the Format Panel.

Blender does *not* add an extension to the file name; you must enter one if you wish.

Previous: Manual/PartI/The Vital Functions	Contents	*Next:* Manual/PartI/Your_First_Animation_in_30_plus_30_Minutes_Part_II

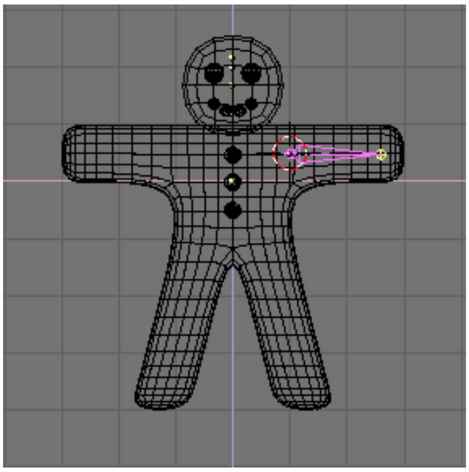
User Manual: Contents | Guidelines | Blender Version **2.4x**

Your First Animation in 30 plus 30 Minutes Part II

If we were going for a still picture, our work up to this point would be enough, but we want Gus to move! The next step is to give him a skeleton, or Armature, which will move him. This is called the fine art of rigging. Gus will have a very simple rigging: four limbs (two arms and two legs) and a few joints (no elbows, only knees), but no feet or hands.

Rigging

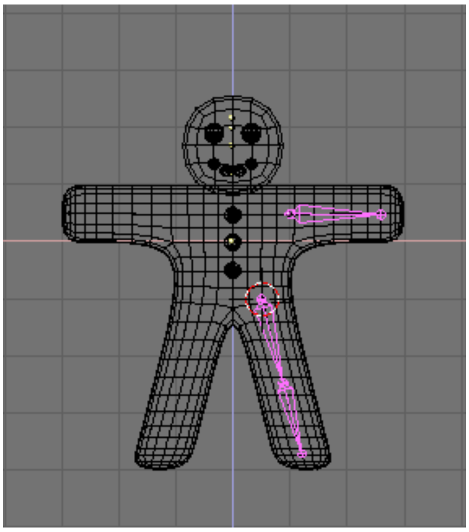
To add the rigging:



Adding the first bone, an elbowless arm.

- Set your 3D cursor where Gus's shoulder is, and press SPACE>>Add>>Armature. A rhomboidal object will appear, which is a bone of the armature system. Enter Edit mode. The end of the bone is selected (yellow).
- Place the other end of the armature in Gus's hand by grabbing (G) and moving the end, (*Adding the first bone, an elbowless arm*). We don't need any other bones right now. You should now have one bone running from the shoulder to the hand area. As you move the end, you will notice that the whole bone gets bigger; you really are scaling up the bone.

 Your First Animation in 30 plus 30 Minutes Part II

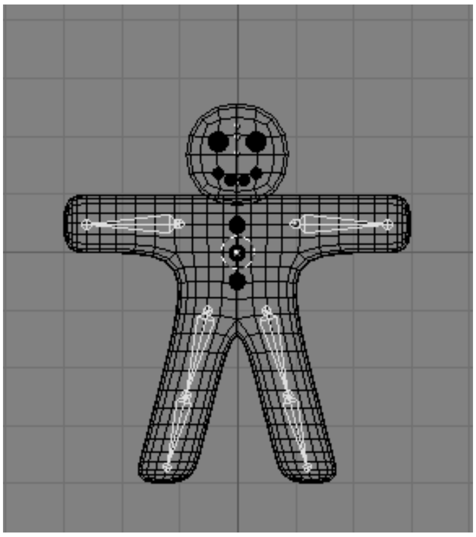


Adding the second and third bones, a leg bone chain.

- Stay in Edit mode, then move the cursor to where the hip joint will be and add a new bone (SPACE>>Add>>Bone).
- Grab (G) and move the yellow end of the new bone to the knee area.
- Now "chain" a new bone from the knee to the foot by Ctrl LMB in the area of the foot. A new "chained" bone will appear automatically linked with the knee and ending at the foot, (*Adding the second and third bones, a leg bone chain*). Another way of "chaining" the new bone would be to extrude using the (E). This variation creates the new bone and places you in grab mode automatically. Use the LMB to then place the bone.

Bone position

The bones we are adding will deform Gus's body mesh. To produce a neat result, try to place the bone joints as shown in the illustrations. We now have three bones that make up Gus's Armature.

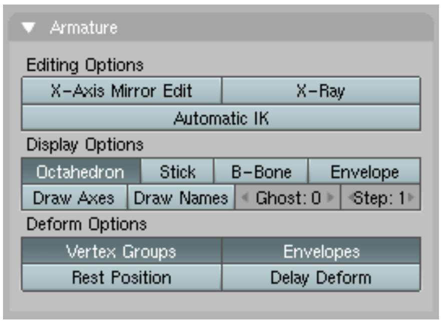


The complete armature after duplicating and flipping.

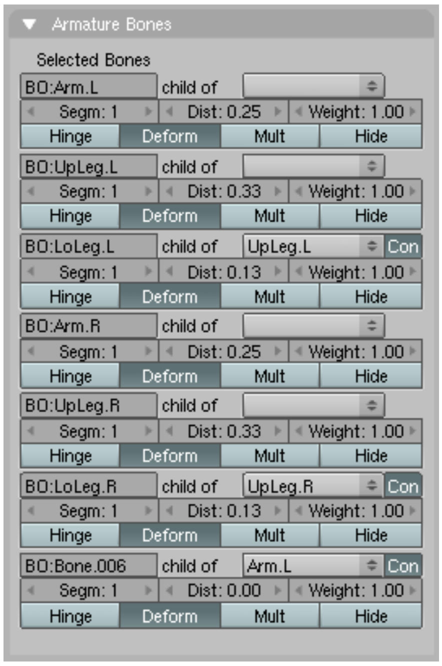
- Now place the cursor in the center and select all bones with A. Duplicate them with Shift D and exit grab mode with ESC. Make sure the cursor is selected as the rotation/scaling pivot. Flip the bones along the X axis relative to the cursor with CTRLM and then X. You end up with (*The complete*

armature after duplicating and flipping).

Once you've selected all of the bones (A), the Edit Buttons window should show an Armature Panel and Armature Bones Panel which contains the Armature controls (*Armature panel* and *Armature Bones panel*).



Armature panel.



Armature Bones panel.

Press the Draw Names button to see the names of the bones in 3D View, then LMB click on the names in the Edit Button window to change them to something appropriate like Arm.R, Arm.L, UpLeg.R, LoLeg.R, UpLeg.L and LoLeg.L, see (*The Edit Buttons window for an armature*). Exit EditMode with (TAB).

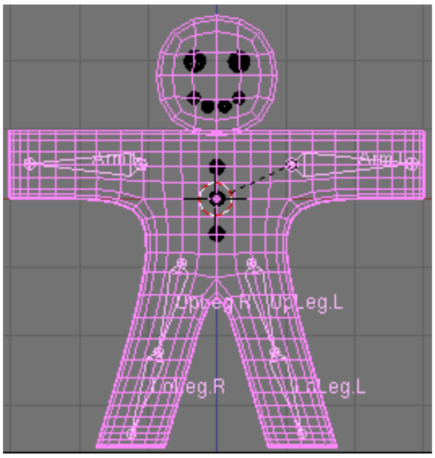
Naming Bones

It is very important to name your bones with a trailing '.L' or '.R' to distinguish between left and right ones, so that the Action editor will be able to automatically flip your poses.

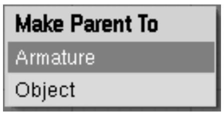
Skinning

Now we must make it such that a deformation in the armature causes a matching deformation in the body. We do this with *Skinning*, which assigns vertices to bones so that the former are subject to the latter's movements.

- In ObjectMode, select Gus's body, then SHIFT select the armature so that the body is magenta and the armature is light pink.
- Press Ctrl P to parent the body to the armature. The (*Parenting menu*) will appear. Select the Armature entry.



Armature parented.

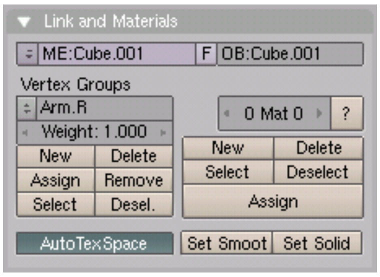


Parenting menu.

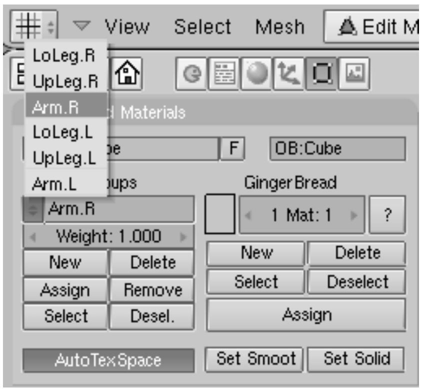
Create Vertex Group menu.

- A new menu appears, asking if you want Blender to do nothing, create empty vertex groups, or create and populate vertex groups (*Create Vertex Group menu*). The last option is considered *automatic skinning*.

- We'll use the automatic skinning option. Go ahead and select Create From Closest Bones. Now select just Gus's body and switch to EditMode (TAB). Notice in the Edit Buttons Window (F9) the presence of the "Vertex Groups" menu and buttons in the Link and Materials Panel, (*The vertex groups buttons in the Edit Buttons window*).

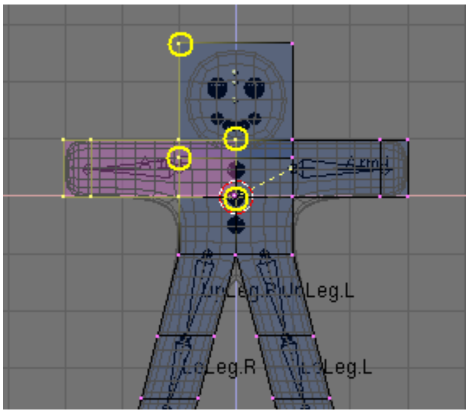


The vertex groups buttons in the Edit Buttons window



The menu with the vertex groups automatically created in the skinning process.

By pressing the Menu Button a menu, with all available vertex groups, pops up −−six in our case. But a truly complex character, with hands and feet completely rigged, can have tens of them! See (*The menu with the vertex groups automatically created in the skinning process*). The buttons Select and Deselect show you which vertices belong to which group.



Gus in EditMode with all the vertices of group Arm.R selected.

Select the Right arm group (**Arm.R**) and, with all vertices de−selected (A, if needed) press Select. You should see something like (*Gus in EditMode with all the vertices of group Arm.R selected*).

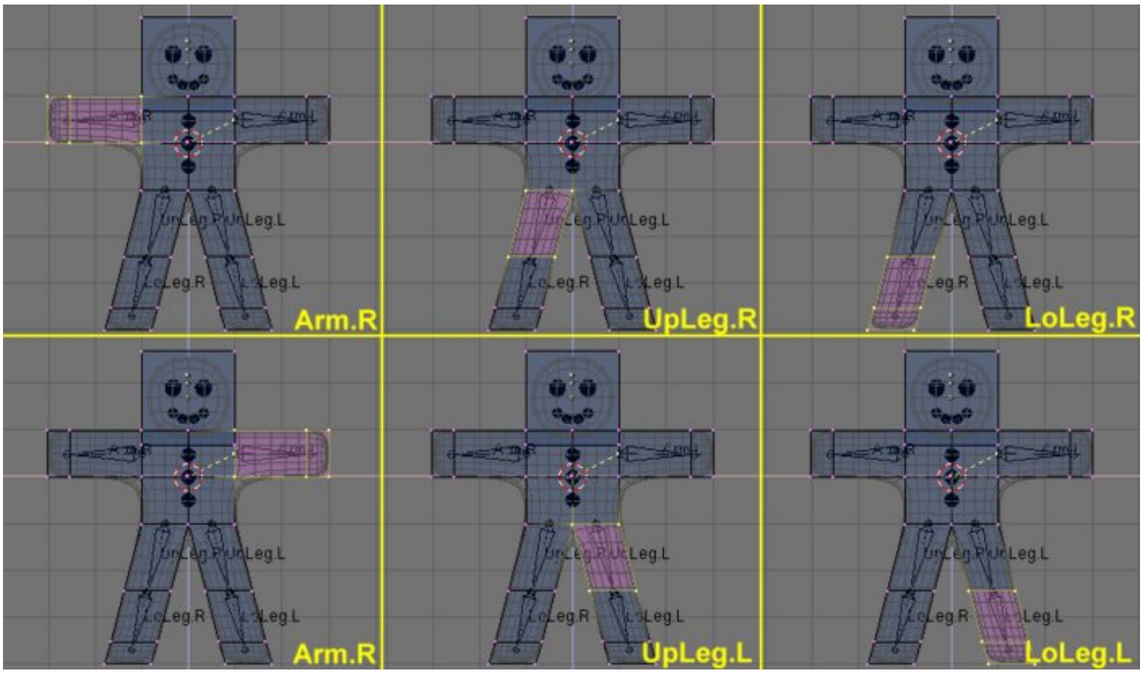
If you don't see the same thing then you probably placed the bones in just the right place such that the *auto skinning* process did a better job of matching vertices with bones. It is highly unlikely that the skinning process matched the vertices to the bones as exactly as you may expect. This requires that you begin to manually adjust the grouping as described in the following sections.

The vertices marked with yellow circles in (*Gus in EditMode with all the vertices of group Arm.R selected*) belong to the deformation group, however, they should not.

The *auto skinning* process found that they were very close to the bone so it added them to the deformation group. We don't want them in this group since some are in the head and some are in the chest, adding them to the deformation group would deform those body parts as well.

To remove them from the group, deselect all the other vertices, those which should *remain* in the group using Box selection (B), but use MMB, not LMB, to define the box, so that all vertices within the box become deselected.

Once the 'undesired' vertices are selected, press the Remove button (*The vertex groups buttons in the Edit Buttons window*) to eliminate them from group (**Arm.R**). Deselect all (A) then check another group. Check them all and be sure that they look like those in (*The six vertex groups*).



The six vertex groups.

Vertex groups

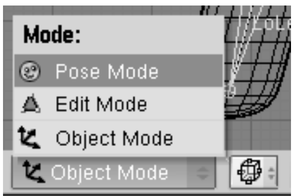
Be very careful when assigning or removing vertices from vertex groups. If later on you see unexpected deformations, you might have forgotten some vertices, or placed too many in the group. You can modify your vertex groups at any time.

Other details

Our deformations will affect only Gus's body, not his eyes, mouth, or buttons, which are separate objects. While this is not an issue to consider in this simple animation, it's one that must be taken into account for more complex projects, for example by parenting or otherwise joining the various parts to the body to make a single mesh. (We'll describe all of these options in detail in later Chapters).

Posing

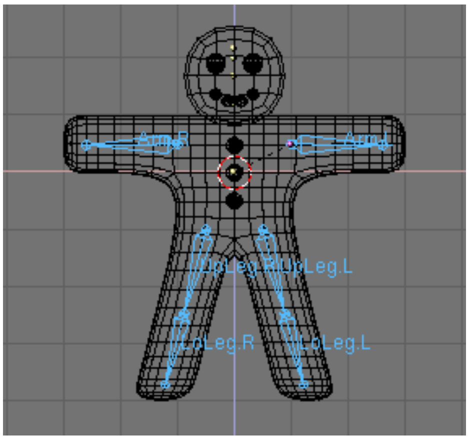
Once you have a rigged and skinned Gus you can start playing with him as if he were a doll, moving his bones and viewing the results.



Mode menu in the 3D Window header.

- Select the armature only, then select Pose Mode from the "Mode" Menu (*Mode menu in the 3D Window header*). This option only appears if an armature is selected.

- The armature will turn blue. You are now in Pose Mode. If you now select a bone it will turn cyan, not pink, and if you move it (G), or rotate it (R), the body will deform!



You are in pose mode now!

Original position

Blender remembers the original position of the bones. You can set your armature back by pressing Alt R to clear the rotation and Alt G to clear the location. Alternatively, the Rest Position button may be used to temporarily show the original position.

Inverse Kinematics

Inverse Kinematics (IK) is where you actually define the position of the *last* bone in the chain, often called an "*End Effector*". All the other bones assume a algorithmic position, automatically computed by the *IK* solver, to keep the chain without gaps (i.e. IK will mathematically solve the chain positions for us). This allows a much easier and precise positioning of hands and feet using IK.

Forward Kinematics

While handling bones in Pose Mode notice that they act as rigid, inextensible bodies with spherical joints at the end. You can grab only the first bone of a chain and all the others will follow it. All subsequent bones in the chain cannot be grabbed and moved, you can only rotate them, so that the selected bone rotates with respect to the previous bone in the chain while all the subsequent bones of the chain follow its rotation.

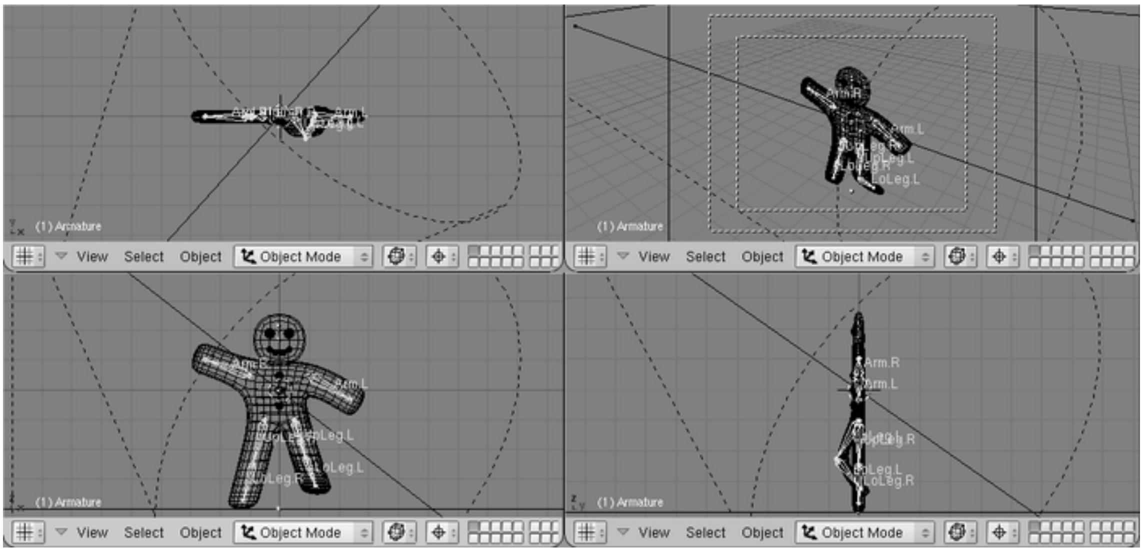
This procedure, called *Forward Kinematics (FK)*, is easy to follow but it makes precise location of the last bone in the chain difficult.

We'll make Gus walk, using FK, by defining four different poses relative to four different stages of a stride. Blender will do the work of creating a fluid animation.

The current frame Num Button in the Buttons window Toolbar.

- First, verify that you are at frame **1** of the timeline. The frame number appears in a NumButton on the right of the Buttons Window Toolbar (*The current frame Num Button in the Buttons window Toolbar.*). If it is not set to **1**, set it to **1** now.

- Now, by rotating only one bone at a time (R), we'll raise **UpLeg.L** and bend **LoLeg.L** backwards while raising **Arm.R** a little and lowering **Arm.L** a little, as shown in *Our first pose.*.

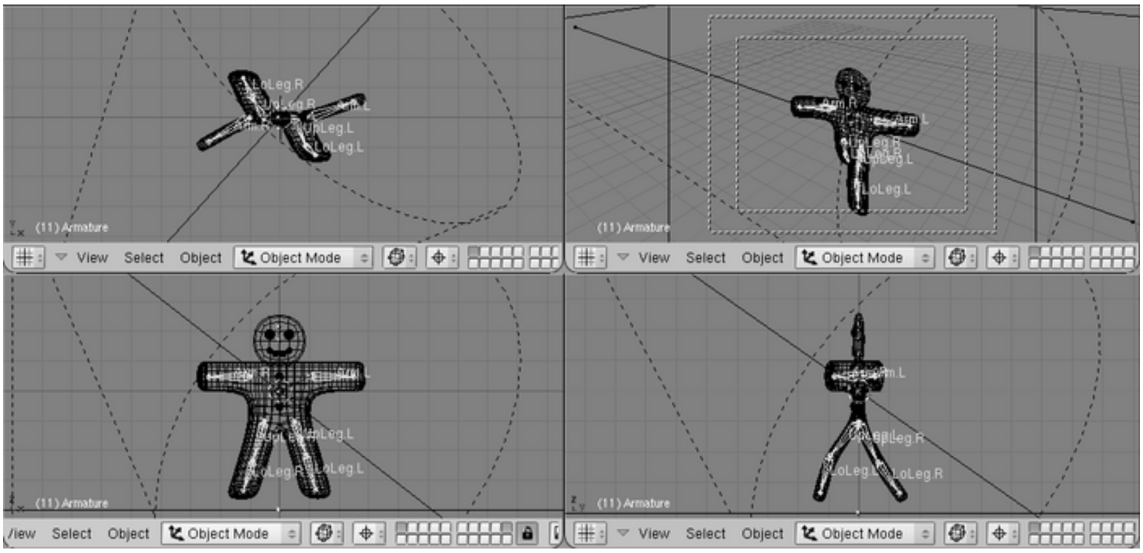


Our first pose.

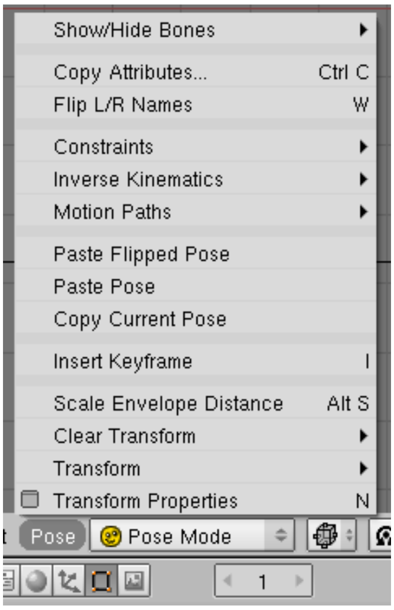
Storing the pose to the frame.

- Select all bones with A. With the mouse pointer on the 3D Window, press I. A menu pops up (*Storing the pose to the frame*). Select LocRot from this menu. This will get the position and orientation of all bones and store them as a pose at frame 1. This pose represents Gus in the middle of his stride, while moving his left leg forward and above the ground.

- Now move to frame 11 either by entering the number in the NumButton or by pressing UPARROW. Then move Gus to a different position, like (*Our second pose*), with his left leg forward and right leg backward, both slightly bent. Gus is walking in place!



Our second pose.



Pose menu

- Select all bones again and press I to store this pose at frame 11.

- We now need a third pose at frame 21, with the right leg up, because we are in the middle of the other half of the stride. This pose is the mirror of the one we defined at frame 1. Therefore, return to frame 1 and, in the Pose Menu in the 3D Window header select the Copy Current Pose entry, see (*Pose menu*). You have now copied the current pose to the buffer.

- Go to frame 21 and paste the pose with the Paste Flipped Pose option in the Pose Menu, see (*Pose menu*). This button will paste the cut pose, exchanging the positions of bones with suffix "**.L**" with those of bones with suffix "**.R**", effectively flipping it!

The pose is there but it has not been stored yet! You must press I with all bones selected.

- Now apply the same procedure to copy the pose at frame 11 to frame 31, also flipping it.

- To complete the cycle, we need to copy the pose at frame 1 *without* flipping to frame 41. Do so by copying it as usual, and by using the Paste Pose entry. End the sequence by storing the pose with I.

Checking the animation

To preview your Animation, set the current frame to 1 and press **ALT−A** in the 3D window.

Gus walks!

The single step in−place is the core of a walk, and once you have defined one there are techniques to make a character walk along a complex path. But, for the purpose of our Quick Start, this single step in−place is enough.

- Change to the Rendering Buttons (F10) and in the Anim panel, below the PLAY button, set the start frame (Sta:) to 1 (it is usually set to 1 by default so you probably won't need to change it) and set the end frame (End:) to 40 (it is set to 250 by default) (*Setting the Rendering Buttons for an animation*). Because frame 41 is identical to frame 1, we only need to render frames from 1 to 40 to produce the full cycle.

Setting the Rendering Buttons for an animation.

- Select AVI Raw as the file type in Format Panel (*Setting the Rendering Buttons for an animation.*). While this is generally not the best choice, mainly for file size issues (as will be explained later on), it is fast and it will run on any machine, so it suits our needs. (You can also select AVI Jpeg to produce a more compact file. However, it uses lossy JPEG compression and will produce a movie that some external players might not be able to play).

Finally, press the ANIM button in Anim Panel. Remember that *all* the layers that you want to use in the animation must be shown! In our case, these are layers 1 and 10.

Stopping a Rendering

If you make a mistake, like forgetting to turn layer 10 on, you can stop the rendering process with the **ESC** key.

Our scene is pretty simple, and Blender will probably render each of the 40 images in a few seconds. Watch them as they appear.

Stills

Of course you can always render each of your animation frames as a still by selecting the frame you wish to render and pressing the **RENDER** button.

Once the rendering is complete you should have a file named 0001_0040.avi in a render subdirectory of your current directory −−the one containing your .blend file.

You can play this file directly within Blender by pressing the Play button beneath the ANIM button (*Setting the Rendering Buttons for an animation*). The animation will automatically cycle. To stop it press ESC. We have produced only a very basic walk cycle. There is much more in Blender, as you'll soon discover!

Previous: Manual/PartI/Your_First_Animation_in_30_plus_30_Minutes_Part_I	Contents	*Next:* Manual/The_Vital_Functions

User Manual: Contents | Guidelines | Blender Version **2.4x**

Opening Files

Mode: All Modes

Hotkey: F1

Menu: File Open

Description

Blender uses the .blend file format to save nearly everything: Objects, Scenes, Textures, and even all your user interface window settings.

Warning

Blender expects that you know what you are doing! When you load a file, you are **not** asked to save unsaved changes to the scene you were previously working on, completing the file load dialog is regarded as being enough confirmation that you didn't do this by accident. **Make sure that you save your files.**

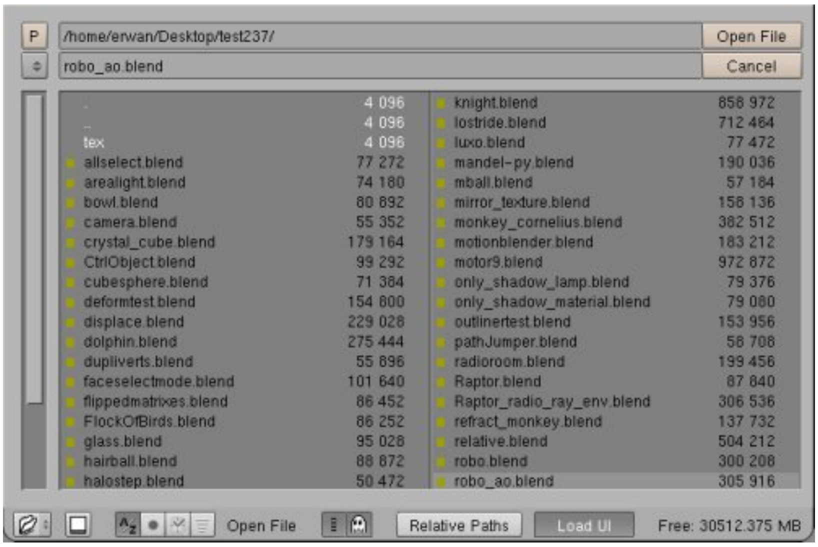
Options

To load a Blender file from disk, press F1. The window underneath the mouse pointer then temporarily becomes the File Selection window as shown in (*File Selection Window − loading.*). The bar on the left can be dragged with LMB for scrolling. To load a file, select it with LMB and then press Enter, or click the Open File button. A file can also be loaded by using the MMB over the name of the file you want. It is then loaded up as if you had pressed enter.

Loading the UI

Inside each Blend file, Blender saves the user interface layout − the arrangement of screen layouts. By default, this saved UI is loaded, over−riding any user defaults or current screen layouts that you have. If you want to work on the blend file using your current defaults, start a fresh Blender, then open the file selector (F1). Turn off the "Load UI" button, and then open the file.

Navigating your Hard Disk



File Selection Window − loading.

The upper text box displays the current directory path, and the lower text box contains the selected filename. (P) moves you up to the parent directory. The button beneath, with the up and down arrow, maintains a list of recently used paths and on the windows platform a list of all drives (C:, D:, etc.). The breadcrumb files (. and ..) refer to the current directory and upper−level directory, respectively.

Other File Open Options

Open Recent
: Lists recently used files. Click on one to load it in.

Recover Last Session

If autosave is turned on, this attempts to load the last hot backup for you. If the file cannot be found, you might need to manually go to your temp directory and find it.

Saving Files

Mode: All Modes

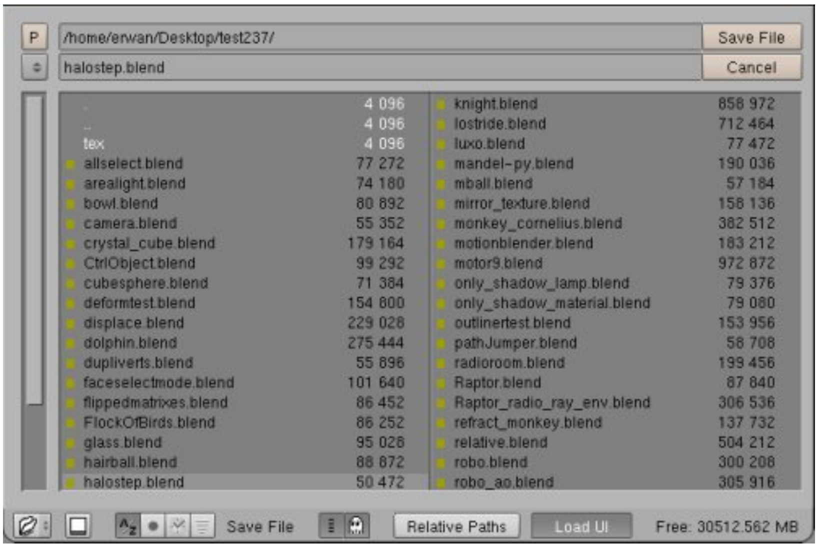
Hotkey: F2

Menu: File Save

Description

Saving files is like loading files. When you press F2, the window underneath the mouse pointer temporarily changes into a File Selection Window, as shown in (*File Selection Window − saving.*).

Options



File Selection Window − saving.

Click the lower edit box to enter a filename. If it doesn't end with "**.blend**," the extension is automatically appended. Then press Enter or click the Save File button to save the file.

If a file with the same name already exists, you will have to confirm that you want to save the file at the overwrite prompt.

Depending on the number of save versions you have set, all existing files with the same name will be rotated to a .blendX file extension, where X is 1, 2, 3 etc. So, if you were working on MyWork.blend, and saved it, the existing MyWork.blend is renamed to MyWork.blend1, and a new MyWork.blend is saved. This way, you have hot backups of old saved versions that you can open if you need to massively undo changes.

Compress Files

Enable File−>Compress Files to squash large files, removing dead space.

Hints

The save dialog contains a little feature to help you to create multiple versions of your work: Pressing NumPad + or NumPad − increments or decrements a number contained in the filename. To simply save over the currently loaded file and skip the save dialog, press Ctrl W instead of F2 and just confirm at the prompt.

Screenshots

In order to facilitate teamwork and rapid prototyping, you might want to quickly take a picture of your window or entire blender window setup.

File −> Screenshot Subwindow takes a picture of your last active window and saves it as a JPG. A window opens, allowing you to specify the location and name of the file.

File −> Screenshot All takes a picture of the entire Blender window.

Rendering

Mode: All Modes

Panel: Render Context Render

Hotkey: F12

Menu: Render Render Current Frame

Description

This section will give you only a quick overview of what you'll need in order to render your scene. You'll find a detailed description of all options in [Rendering](#).

Options

The render settings are in the Scene Context and Rendering Buttons Sub−context (*Rendering options in the RenderingButtons.*) which is reached by clicking the , or by pressing F10.

Rendering options in the RenderingButtons.

In the Output panel, the top field contains the path increment (default: "/tmp/") and optionally a filename prefix to use when rendering. The Path Increment is either an absolute address or a relative address. An absolute address is something like "C:\Documents\Blender\" and a relative address is a breadcrumb notation ("./" or "../") meaning to start with the current or parent directory of the Blender installation location, or a double slash ("//") meaning put the file in the directory from where the blend file was loaded.

Invalid Paths

If the construction of the path is illegal and rejected by the operating system, your file can end up in the Blender installation directory, root directory, or some other place.

The Format Panel controls the format of the render. The full size (number of pixels horizontally and vertically) and file format of the image to be created are picked here. You can set the size using the SizeX and SizeY buttons. Clicking the selection box just below the size buttons opens a menu with all available output formats for images and animations, which is currently "Jpeg" in (*Rendering options in the RenderingButtons*).

Now that the settings are complete, the scene may be rendered by hitting the RENDER button in the Render Panel or by pressing F12. Depending on the complexity of the scene, this usually takes between a few seconds

and several minutes, and the progress is displayed in a separate window. If the scene contains an animation, only the current frame is rendered. (To render the whole animation, see Rendering Animations).

If you don't see anything in the rendered view, make sure your scene is constructed properly. Does it have lighting? Is the camera positioned correctly, and does it point in the right direction? Are all the layers you want to render visible? Make sure Blender Internal is chosen in the dropdown box below the RENDER button.

Saving to disk

A rendered image is not automatically saved to disk. If you are satisfied with the rendering, you may save it by pressing F3 and using the save dialog as described in *Saving files*. The image is saved in the format you selected previously in the Format Panel.

Hints

Click the Extensions button in the Scene (F10) Render context Output panel so that Blender will add the type extension (i.e. ".jpg") automatically to image files!

Setting the default scene

If you don't like Blender's default window set−up, or want specific render settings for each project you start, or you want to save your Theme? No problem. You can use any scene file as a default when Blender starts up. Make the scene you are currently working on the default by pressing Ctrl U. The scene will then be copied into a file called .B.blend in your home directory.

You can clear the working project and revert to the default scene anytime through the menu entry File>>New or by pressing Ctrl X. But remember to save your changes to the previous scene first!

Other File Menu Options

Append or Link
: You don't have to load a complete file; you can load in only selected parts from another file if you wish. Appending and Linking is discussed here.

Import
: Blender can use information stored in a variety of other format files which are created by other graphics programs. It does this by running a script to import the file.

Export
: Normally you save your work in a .blend file, but you can export some or all of your work to a format that can be processed by other graphics programs. To do so, you run an export script.

Previous: Manual/The Interface	Contents	*Next:* Manual/User Preferences

User Manual: Contents | Guidelines | Blender Version **2.46**

User Preferences and Themes

Mode: All Modes

In Blender, you can customize your defaults, and once you are satisified, save them via File−>Save user Defaults. If you ever want to start completely over, simply restore "factory" settings via File−>Load Factory Settings

Description

Blender has a few options that are not saved with each file, but which apply to all of a user's files instead. These preferences primarily concern the user interface handling details, system properties like mouse, fonts, and languages.

As the user preferences are rarely needed, they are neatly hidden behind the main menu. To make them visible, pull down the window border of the menu (usually the topmost border in the screen). The settings are grouped into seven categories which can be selected with the violet buttons shown in (*User preferences window*).

Options

Because most buttons are self−explanatory, or display a helpful tool−tip if you hold the mouse still over them, we won't describe them in detail here. Instead, we will just give you an overview of the preference categories and some suggestions.

Pull down the User Preference Window to reveal many customization options. Each section of User Preferences Window displayed in default state.

View & Controls

Settings concerning how the user interface should react to user input, such as which method of rotation should be used in 3D views. Here you can also activate 3−button mouse emulation if you have a two−button mouse. MMB can then be input as Alt LMB.

In particular, I just want to call out the Smooth View setting used in transitioning your 3D window from one view to another (e.g. from Top view to Side view). A higher value (e.g.1000) smooths the transition from view to view, instead of jumping. A very nice effect found in other packages, and is pleasing on the eye.

Edit Methods

Lets you specify the details for the workings of certain editing commands like duplicate. You can also change the amount of undo steps and whether the undo should work globally or locally. For more information on Undo and Redo, click here.

Add New Objects:

Enable Switch to Edit Mode if you want the Blender to automatically go into edit mode when you add an object.
Enable Aligned to View when new objects are added, and they will be automatically rotated to face the view you are in. Otherwise, they will be global axis−aligned.

Auto key framing for animation is also controlled from here.

- Automatic Keyframing Options
    - *Auto−Keying Enabled* Automatically sets keys after a transformation of either objects or bones, removing the need to use the I key.
        - *Add/Replace Keys* The default behavior. It Adds new keys on transformation, and, if a key already existed on the Ipo on that frame, the key is replaced with the new one.
        - *Replace Keys* Will not add new keys to Ipos. This option will only replace existing keys.
    - Other Automatic Keyframing toggles
        - *Available* Only adds keyframes to existing Ipo curves. For example, with this enabled, translation and rotation can be set in the 3D view, but only a translation key will be created if there exists only a translation Ipo, but no rotation Ipo.
        - *Needed* The default behavior for autokeying is to create a key anytime a transformation occurs, in all available channels. With the *Needed* option, values that do not change between the previous and next keys do not receive new keys. In other words, if you move an object along the Y axis only, new keys will not be set for the X or Z axes as their value does not change.
        - *Use Visual Keying* Uses the Visual Keying method for objects and bones that have certain Constraints that can affect the key values. For example, setting a key on an object with a Copy Location constraint would normally set the key for it's unconstrained location. Enabling this option causes the key to be set for the constrained location.

Language & Fonts

International Fonts, when enabled, allow you to use English fonts as well as foreign fonts such as Kanji and Pharsi as the labels for Blender's buttons. When enabled, you must *Select Font* which will load the font file. Optionally, you can change the default font size in picas. Blender's default is a nice, clean sans−serif font, but you can use anything you want, even Wingdings!

Along with International Fonts, you can choose your native language (English by default), and whether Tooltips, Buttons and Toolbox text should be displayed in that language. Use Textured Fonts for better display of characters.

Themes

Blender allows the utilization of Themes to define custom interface colors and icons. You can manage themes from here, and two are built−in: Default and Rounded. Many others are available from the Internet, such as

Dark Alpha and GONX. Each theme is a python script, usually ending in *_theme.py. You can import other people's themes by simply copying their .python script into your Blender/Scripts directory and restarting Blender. Once Blender has restarted, switch to the script window, click "scripts" and choose "themes", then you can select each theme that is there one at a time. Close Blender. When you restart Blender, you will see your themes listed under user preferences −> themes. Finally, press Ctrl U to save it as your default.

To preserve your current theme when opening a new file, disable the Load UI button.

Customizing Theme

Click on a custom theme from the selection box, or Add a copy of the default. When you do, many more columns of buttons will be shown. The second column of buttons, in order:

> **Name** of the theme. Click into the field and change what is there)
> **Section** of the theme to change, mostly organized by window type, and has the tool tip "Specify theme for...". (3D View is the default choice)
> **Element** within that part to customize (Background is the default choice)

The next column over shows the current color for that element. Change it using the RGB sliders, or clicking on the color swatch and using the eyedropper.

Creating / Saving a Theme

File −> Export −>Save Current Theme

Customizing Icons

Blender uses LOTS of icons to represent functions; clicking an icon invokes that function. You can change many icons to suit your preference.

Default

All images in this wiki have been screenshot using the Default icon and theme, so that it all looks consistent and less confusing. If you are looking at a private tutorial, or a forum sceenshot, keep in mind they might have changed all the colors and icons, and you will have to match up based on icon placement.

To use custom icons within Blender, first create an icon file. It is a graphical image, created as a Blender Render or your favorite graphical image manipulation program as a 14x22 pixel image. Create a directory "icons" inside the ".blender" directory in Your Blender's installation. Copy the icon file set there.

In Blender, enter the "Themes" area in the "User preferences" window. Click on a custom theme, or Add a copy of the default. When you do, many more columns of buttons will be shown as described above. In the second column, the top field allows you to change the theme name. The button below it allows you to select what part of the theme to change and has the tool tip "Specify theme for...". Select 'UI and Buttons' from the drop down selection that reads '3D View'. After you did this you can select 'Icon File' from the drop down selection that's named 'Outline' (the Element to change within that . A new selection next to 'UI and Buttons' will appear and you can select your icon set there.Now You can browse icons that are in mentioned in the "icons" directory and use your favourite ones!

Auto Save

The creative process is very involving, and the artist often gets so deep into modeling and animation that he or she often forgets to bathe, eat, and especially save copies of their work. A computer crash, power outage, or simply taking a bad fork in the creative path can result in lost work or corruption of the desired product. Have no fear of immersing yourself, because Blender provides several ways to automatically save backup copies of work in progress. This sub−panel allows you to configure the two ways that Blender allows you to regress to a prior file version.

Save Versions

The "Save Versions" button tells Blender, when you *manually* select File/Save, to save the specified number of previous versions of your file. In your current working directory, these files will be named .blend, .blend1, .blend2, etc. on up to the number of versions you specify, with the older files named with a higher number. Typically, 9 versions are more than sufficient.

Auto Save Temp Files

Clicking the "Auto Save Temp Files" button tells Blender to *automatically* save a hot backup copy of your work−in−progress to the temp directory. Selecting this button reveals two more buttons. The first, "Minutes" button specifies the number of minutes between automatic saves. The second "Open Recent" button allows you to open the most−recent auto−save file. The auto−save file is named using a random number, has a .blend extension, and is placed in the Temp directory (refer to the "File Paths" tab). We recommend that you use these buttons to autosave into your temp folder filepath, and set how many minutes go between automatic saves (5 to 10 minutes is sufficient).

Then, when you have done something terrible to your beautiful model, you have four choices: 1) keep working forward and try to cover up or build on your accident, 2) undo with Ctrl Z, 3) regress to (open) a previously saved version in your working directory, or 4) regress to the prior auto−saved version (which is where the next button comes in). To regress to the last auto−save version, simply click the "Open Recent") button and the most recently saved work−in−progress version from the Temp directory will be loaded.

Warning

Clicking the "Open Recent" button will immediately load the most recent save, and you will lose any changes that you have made in the intervening minutes.

Upon loading the Temp version, you may File/Save it over the current file in your work directory as a normal .blend file.

Exit No Save

When you close Blender and exit the program, Blender does not prompt you to save your work if changed. However, it automatically saves the current work−in−progress in a file called "quit.blend" in your Temp directory. If you realize that you have forgotten to save before exiting Blender, simply manually navigate to the Temp directory and open the "quit.blend" file, and save it over your work file.

Recent Files

When you use File −> Open Recent, this control specifies how many recent files to keep track of, and how many will appear in that list.

Save Preview Images

TBD.

System & OpenGL

Solid OpenGL lights

When working in solid view, you can "light" the objects in your workspace with up to three light sources. The direction of each source is set by dragging inside the light sphere, and the colors of the diffuse and specular shading is set by clicking on the color swatch and using the popup color picker

Auto Run Python Scripts

When enabled, selecting a script from any menu runs that script. However, Python is a powerful language and If you suspect a virus or get a blend file from an untrustworthy source, disable this prior to opening the blend file to prevent the script from running.

Win Codecs

Codecs are routines that encode and decode video streams. Enable this to use any codec found in your system. Some codecs can mis−behave, and can falsely assert themselves. If you have one of these pontificating codecs on your system, disable this.

Color range for weight paint

When weight painting, by default we use a colorful band of color to represent weights ranging from 0.0 to 1.0. If you want to use a different set of colors to represent the range of values when weight painting, click *ColorBand* and use the colorband control to set the colors that correspond to the range.

Audio Mixing Buffer

When mixing audio or using audio in the game engine, this allows you to set aside memory for sound.

Verse

Verse allows multiple users to work on the same blend file. Enter the URL of the Master and your username here. Then use the File−>Verse menu to get connected to your buddies.

Keyboard

if you don't have a numerical keypad and want to emulate it (for laptops). if your fat fingers keep pressing the darn caps lock instead of the tab key, disable it.

System

Prefetch allows Blender to get anticipated frames into memory, allowing you work smoother. If you are using the Video Sequence Editor a lot, you might want to increase your MEM Cache Limit. If you are rendering frames on request as a slave, enter the port number to listen for requests on.

OpenGL

Allow you to fine−tune how OpenGL displays textures and functions on your system.

File Paths

Choose the default paths for various file load dialogs. Remember that the // at the beginning of a pathspec means "where the .blend file is currently saved". "/" at the beginning means the top directory of the currently active drive.

Relative Paths Default

This button is at the top right of the panel. If you enable this button, the internal path to any ancillary file, such as an image texture, will be saved, not as an absolute path starting with the drive letter, but as a pathspec to the file starting with the location of the .blend file. This enables you to zip and move entire projects from one PC to another, and all ancillary files will be found, even if on PC A you save the .blend on drive C: and on PC B you save the .blend on drive D:. Use this feature if you think you

will have to be sharing or working on the project across multiple machines.

YFexport
: Blender communicates with YafRay by exporting an XML file. This entry tells Blender where to save it. Suggestion: **C:\tmp**

Fonts
: Where to find TrueType Fonts. Suggestion: **C:\Windows\Fonts**

Render
: Where to put rendered output. Suggestion: **//render**

Textures
: Where to find pictures and images for texturing surfaces. Suggest: **C:\Blender\lib\tex** where **lib** is your local library. As an alternative, you can use a local copy of the textures relative to your project. This is handy if you are going to be moving the .blend file from machine to machine and want to pack the textures. In this case, you would want to enter **//textures** to mimic the folder that is created when you unpack textures using the *write files to current directory* option.

Python
: Blender's customization scripting language and functionality extensions. If left blank, Blender uses the distributed scripts which are located in your install directory under *Blender\.blender\scripts* directory. Suggestion: **C:\Blender\scripts**. The **Python** pathspec should NOT end in a slash; this is an exception. If you use a local library/repository of scripts, you should remember to refresh it with the latest distributed scripts when you upgrade Blender.

Tex Plugins
: Plugin DLL's to augment texturing. Suggest: **C:\Blender\bin\plugins\texture**

Sounds
: wave files for soundtracks and sound effects. Suggest: **C:\Blender\lib\wav**

Seq Plugins
: When using the Video Sequence Editor within Blender, Blender has a host of nifty effects that can be augmented by DLLs. Suggest: **C:\Blender\bin\Plugins\sequence**

Temp
: The generic trash can, where your exist session save file is saved (quit.blend) as well as autosave files. Suggest: **C:\tmp**

You can manually enter a path, or LMB click the little file folder icon to the right of the field to use Blender's file browser to navigate you hard drive/network. Doing this is recommended in order to avoid typo's. The folder searcher puts in the pathspec ending in a slash.

If any folder name changes, or the folder is moved or deleted, you will have to come back here and change them again, since Blender has no way of being informed of those changes.

Saving your Preferences

When you press Ctrl U, you will save a file called .B.blend in the .blender folder underneath your Blender installation that contains the present setup, including all screens and scenes. Please note that because of its weird filename, Windows OS's may try to hide it from you. Also, it might be saved in an Application Data directory specific to your User Profile. If it is not saving your changes, be sure you have security rights set to allow changes to files in the folder; this is especially the case with Microsoft Vista OS, as it definately does not by default allow any program to change any file in the *Program Files* directory.

In any event, it is a plain old .blend file; so if you have objects etc in your file when you Ctrl U, those will also be the default the next time you start.

If the file is lost or accidentally deleted, Blender will re−create it on the next startup.

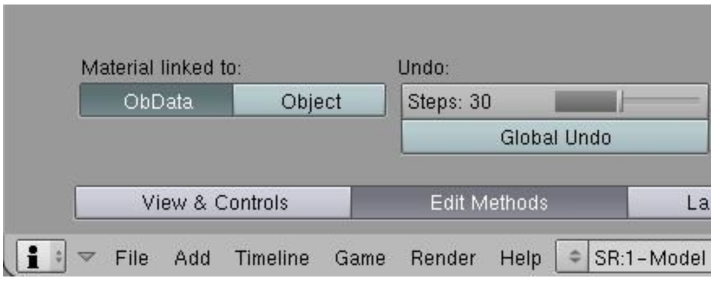
Previous: Manual/The Vital Functions Contents *Next:* Manual/Undo and Redo

User Manual: Contents | Guidelines | Blender Version **2.44**

Description

Blender has many options and features to make sure that you do not lose your work. First, it saves your actions in a list. At any time, you can tell Blender to back up in the list and undo most recent changes. Second, when you start Blender, one of the File options is to Recover Last Session. When you exit Blender, it saves the current file in a quit.blend file; Recover Last Session merely loads that file back in. Third, you can tell Blender via User Preferences to automatically save versions "behind the scenes", and to keep old copies of your entire files every time you do manual saves.

Getting Started



By default, Undo is not turned off although it takes precious memory. To enable or disable undo, drag down your User Preferences window and click Edit Methods. In that panel, you may set:

- Steps: This numeric slider sets how many steps, or actions, to save. If you set this to 30, you will be able to undo the last 30 actions that you performed.
- Global Undo: This enables Blender to save actions outside of some mesh editing actions, for example, moving individual vertices while a mesh is in one editing session; each vertex move can be undone.

Undo

Mode: All Modes

Hotkey: Ctrl Z

When you have done something terrible to your beautiful model, you have the following choices:

1. keep working forward and try to cover up or build on your accident,
2. Undo (via Ctrl Z),
3. Revert to (open) a previously saved version in your working directory, or
4. Regress to the prior auto−saved version (if you have AutoSave turned on)

To regress to the last auto−save version, simply move your cursor toward the top of your screen until it hovers over the boundary of the User Preferences header and changes to an Up−Down arrow. Click and drag that header down to reveal the User Preferences window. Click the AutoSave tab, and click the "Open Recent" button. Immediately, the most recently saved work−in−progress version from the Temp directory will be loaded.

Warning

Clicking the "Open Recent" button will immediatly load the most recent save, and you will lose any changes that you have made in the intervening minutes.

Upon loading the AutoSaved version, you may File/Save it over the current file in your work directory as a normal .blend file.

If you have versioning turned on (again specified in the User Preferences − AutoSave tab), each time you File−>Save your .blend file, Blender pushes down the previous save to a .blend1 file, and the .blend1 to a .blend2 file, and so on up to the max number of versions that you have specified. To revert to one of those files, simply File−>Open and select one of those .blendx files. Then File−>Save it to bring it to the top of the stack.

Redo

Mode: All Modes

Hotkey: Shift Ctrl Z or Ctrl Y

Just as Ctrl Z undoes an action, Shift Ctrl Z re−does the last undone action(s).

History

Mode: All Modes

Hotkey: Alt U

Alt U displays the Global History of what you have done as a list of actions named generally for what you did. Clicking on any action reverts you back to that state just before the next action was performed.

Previous: Manual/The Vital Functions	Contents	*Next:* Manual/Library and Data System

User Manual: Contents | Guidelines | Blender Version **2.43**

Overview

Each .blend file contains a database. This database contains all scenes, objects, meshes, textures, etc. that are in the file. A file can contain multiple scenes and each scene can contain multiple objects. Objects can contain multiple materials which can contain many textures. It is also possible to create links between different objects.

Mode: All Modes, Any Window

Hotkey: Shift F4 − Datablock Browser

To access the database, press Shift F4 and the window will change to an Datablock browser window, which lists the Objects in your .blend file. To go up a level, click the breadcrumbs (..) and then you will see the overall structure of a file: Action, Armature, Brush, Camera, Curve, Group, ... and so on (including Objects).

LMB Selecting any datablock type, Mesh, for example, will give you a listing of the Meshes used in the file, along with how many users there are for that class. For example, if you had a car mesh, and used that car mesh for six cars in a parking lot scene, the Mesh listing would show the Car and then the number 6.

Mode: Data Select Browser

Hotkey: F − Fake User

RMB Selecting certain kinds of datablocks (Materials, Images, Textures...) and pressing F will assign a Fake user to those datablocks. With a fake user in place, Blender will keep those datablocks in the file, even if they have no 'real' users. Datablocks without a user, real or fake, are not saved in the .blend file. Pressing F again toggles the Fake−user assignment. Performing this action is the same as clicking the **F** button next to Material names, Image names, etc.

Outliner and OOPS Schematic

You can easily inspect the contents of your file by using the Outliner window. This window displays the Blender data system. (Fully documented here.) This window offers two views of the database. The Outline view allows you to do simple operations on the objects. These operations include selecting, renaming, deleting and linking. The OOPS (Object−Oriented Programming System) Schematic view allows you to easily see how datablocks are linked. You can filter the view by using buttons found in the header.

Users (Sharing)

Many datablocks can be shared among other datablocks; re−use is encouraged. For example, suppose you have a material for one object, named "Glossy". You can select a second object, for example, one that does not have a material yet. Instead of clicking ADD NEW for the material, click the little up−down arrow next to the ADD NEW, which brings up a list of existing materials. Select "Glossy". Now, these two objects share the same Material. You will notice a "2" next to the name of the material, indicating that there are two users (the two objects) for this material. Other common examples include:

Sharing textures among materials

Sharing meshes between objects ("clones")
Sharing IPO curves between objects, for example to make all the lights dim together.

Fake User

Blender removes all datablocks that have not been linked to anything when you open the file. Because of this, sometimes you may find it useful to link unlinked datablocks to a "fake user". You can do this by hitting the F button next to the name of the datablock.

Copying and Linking Objects Between Scenes

Sometimes you may want to link or copy objects between scenes. This is possible by first selecting objects you want to link or copy and then using the "Make Links" and "Make Single User" items found in Object menu in the 3D viewport header. Use "Make Links" to make links between scenes. To make a plain copy, you first make a link and then use "Make Single User" to make a stand−alone copy of the object in your current scene. Further information on working with Scenes can be found here.

Appending or Linking Across Files

The content of one .blend file is easily accessed and put into your current file by using the File Append function (accessed at any time by Shift F1). To find out more about how to copy or link objects across .blend files, click here.

Proxy Objects

Proxy Objects allow you to make (parts of) Linked data local. For example, this allows an animator to make a local 'copy' of the handler bones of a character, without having the actual rig duplicated. This is especially useful for character animation setups, where you want the entire character to be loaded from an external library, but still permit the animator to work with Poses and Actions. Another example: you can have a modeler working on the shape (mesh) of a car and another painter work on the materials for that car. The painter cannot alter the shape of the car, but can start working with color schemes for the car. Updates made to the shape of the car are applied automatically to the painter's proxy.

See also this for more useful information about the database system.

Pack and Unpack Data

Blender has the ability to encapsulate (incorporate) various kinds of data within the .blend file that is normally saved outside of the .blend file. For example, an image texture that is an external .JPG file can be put "inside" the .blend file via File Pack Data. When the .blend file is saved, a copy of that .JPG file is put inside the .blend file. The .blend file can then be copied or emailed anywhere, and the image texture moves with it.

You know that an image texture is packed because you will see a little Christmas present gift box displayed in the header.

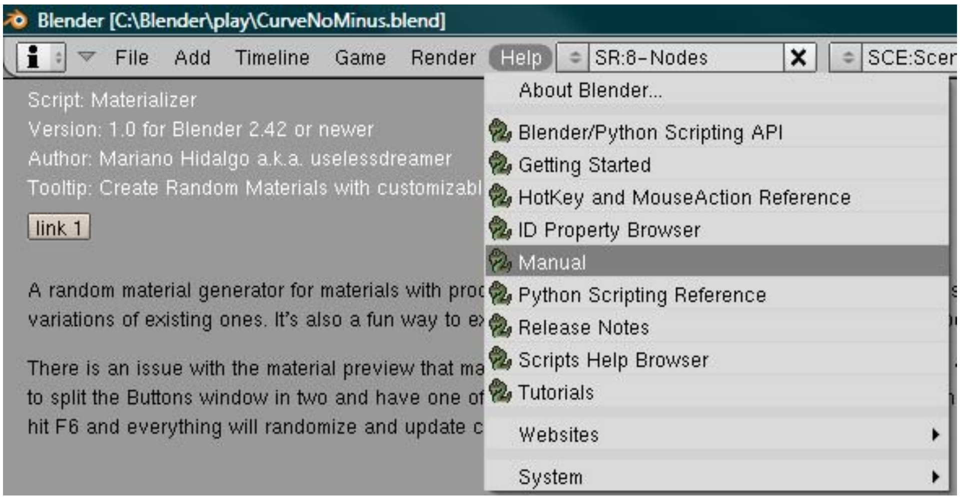
Unpack Data

When you have received a packed file, you can File UnPack Data. You will be presented with the option to create the original directory structure or put the file in the // (directory where the .blend file is). Use "original locations" if you will be modifying the textures and re−packing and exchanging .blend files, so that when you send it back and the originator unpacks, their copy of the textures will be updated.

Previous: Manual/Undo and Redo	Contents	*Next:* Manual/Help

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.44**

Blender has an interesting mix of built−in, loadable, and web−based help, all accessed with Blender. All of the help is accessed through the Help menu at the top of the page in the User Preferences window header. Of course, any web−pages can be saved to your local hard disk or printed using your web browser for off−line viewing. We use web−based help so that we can bring you the latest up−to−date help.



Enabling Help

Some forms of help start up your web browser and access the Blender Foundation's web servers. In order to do this, you must configure, with your operating system, a web browser as a default. If you have a dial−up connection, you must configure the web browser to automatically dial out when it starts if there is no active internet connection available.

About Blender

Displays the splash screen image, identifying the package and version.

Help Topics

Blender/Python Scripting API

Web−based help pages describing the application programming interface (API) that Blender exposes to Python. Python is a general programming language that can do many things on its own. In order for it to do things inside Blender, it has to access Blender functions, like creating objects, moving them, etc. Python does this by calling on specific Blender API calls, like ob.setLocation to change an object's location.

Getting Started

Web−based help pages listing specific links to official sites maintained with fool−proof, easy tutorials that can help you get started and comfortable using Blender.

HotKey and MouseAction Reference

Internal There's an old adage among Blender users:

"One hand on the mouse, the other on the keyboard."

Blender makes extensive use of hotkeys to allow you to quickly perform common actions, like R to Rotate the selected object. This internal page tells you what all those keys do, separated into Arrows, Letters, Mouse actions, Numbers, etc. It has a Search Function, so if you vaguely know the term for *what* you want to do, you can search for the hotkey to find out *how* to do it.

ID Property Browser

Internal For all the various kinds of objects in your .blend file, this screen allows you to find them (starting at the Scene level) and drill down to inspect their properties.

Manual

Web−based Brings you to the main table of contents page of the wiki.

Python Scripting Reference

same as Blender/Python Scripting API

Release Notes

Web−based A page listing all the Release Notes, telling you what has changed and new features added to Blender, back several versions. The release notes provide you with a quick explanation of "what's new", frequently along with examples and demonstrations and movies/images.

Scripts Help Browser

Internal This window allows you to choose a script (from your \scripts directory) by category, and shows you the help information embedded within the script. Use this to find out more about what these custom Python scripts do, and how to use them.

Tutorials

'*Web−based* Lists simple, step−by−step tutorials that can provide you with easy−to−follow directions on how to use Blender. Please also visit the wiki Tutorials page.

Websites

Web−based Lists frequently accessed websites related to Blender and its development.

System

With a 3D View window active, clicking Benchmark enables you to benchmark three different actions in Blender. The resulting statistics, particularly the 'operations per second (ops/s) is useful for comparing the performance of Blender across different machines. For a more robust benchmark, refer to the [official Benchmark site.]

System Information creates an info.txt file which lists various key properties of your system and Blender, useful in diagnosing problems.

Previous: Manual/Library_and_Data_System	Contents	*Next:* Manual/Interaction_in_3D

Description

Using Blender, you create a world that exists in four dimensions:

1. left−right, commonly called the x axis
2. forward−backward, commonly called the y axis
3. up−down, commonly called the z axis
4. time−sensitive, through animated objects, materials, and motion captured in frames

The problem is, you have a two−dimensional computer screen in front of you! Your mouse can only move left−right and up−down. You cannot go back in time, and you can't literally reach out into the screen and grab an object and move it somewhere else (although I did see a scary movie once where the guy reached through a mirror, but that's another story).

Instead, you have to tell Blender to do it for you. This section tells you how to navigate around in your virtual world using the unique Blender interface.

Previous: Manual/Help	Contents	*Next:* Manual/Navigating_in_3D_Space

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.4x**

Navigating in 3D Space

Mode: All Modes

Description

Blender lets you work in three−dimensional space, but our monitor screens are only two−dimensional. To be able to work in three dimensions, you must be able to change your viewpoint as well as the viewing direction of the scene. This is possible in all of the 3D Viewports. While we will describe the 3D Viewport Window, most other windows use an equivalent series of functions. For example, it is possible to translate and zoom a Buttons Window and its Panels.

Mouse Buttons and Numpad: If you have a mouse with less than three buttons or a keyboard without numpad, please refer to the [Keyboard and Mouse](#) page of the manual to learn how to use Blender with them.

Rotating the View

Mode: All Modes

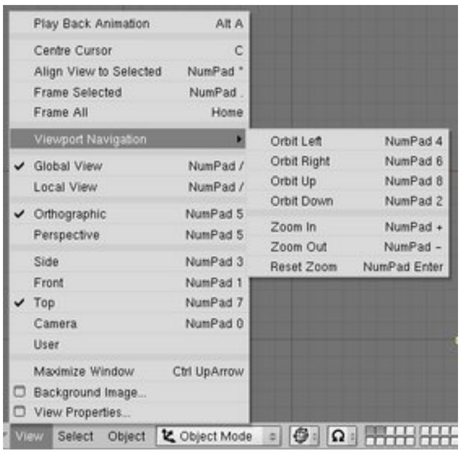
Hotkey: MMB / NumPad 2 / NumPad 4 / NumPad 6 / NumPad 8

Menu: View View Navigation

Description

Blender provides four default viewing directions: Side, Front, Top and Camera view. Blender uses a right−angled "Cartesian" coordinate system with the Z axis pointing upwards, "side" corresponds to looking along the X axis, in the negative direction; "front" along the Y axis; and "top" along the Z axis; The Camera view shows the current scene as seen from the Camera view point.

Options



A 3D Viewport's view menu.

You can select the viewing direction for a 3D Viewport with the View Menu entries (*A 3D Viewport's view menu.*) or by pressing the hotkeys NumPad 3 for "side", NumPad 1 for "front", NumPad 7 for "top". You can select the opposite directions if you hold Ctrl while using the same numpad shortcuts. Finally NumPad 0 gives access to the "Camera" viewpoint.

Hotkeys

Remember that most hotkeys affect the window that has focus, so check that the mouse cursor is in the area you want to work in before your use the hotkeys!

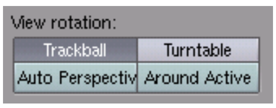
Apart from these four default directions, the view can be rotated to any angle you wish. Click and drag MMB on the Viewport's area: If you start in the middle of the window and move up and down or left and right, the view is rotated around the middle of the window. Alternatively, you can press and hold Alt while dragging LMB in the Viewport's area.

To change the viewing angle in discrete steps, use NumPad 8 and NumPad 2 (which correspond to vertical MMB dragging, from any viewpoint), or use NumPad 4 and NumPad 6 to rotate the scene on the XY plane only.

TrackBall/Turntable

By default, when you rotate the view as described in The viewing direction (rotating) section, you are rotating the scene as though you are rolling your hand across a "**Trackball**". For some users this is intuitive and for others it is not. If you feel you are having difficulties with this style of 3D window rotation you can switch to the "**Turntable**" style.

The "**Turntable**" style is fasioned more like a record player where you have two axes of rotation available and the world seems to have a better definition of what is "Up" and "Down" in the world. The downside to using the "**Turntable**" style is that you lose some flexibility when working with your objects. However, you gain the sense of "Up" and "Down" which can help if you are feeling disoriented. Of course you can always switch between the styles depending on what you are working on.



View rotation

To change the rotation "Style" use the [User Preferences Window](); remember to pull the main window down because only the header shows by default. Click on the "View & Control" button to reveal a page of buttons relating to *Views* and *Control* functionality. You will see an area for choosing the "View rotation:", see (*View rotation*). There are two additional buttons for controlling the display in the 3D window. "Auto Perspective" will automatically switch to Perspective whenever the view is rotated using MMB. "Around Selection" will rotate the view around the center of the current selection. If there is no selection at that moment (ex. if you used A to deselect everything) the last selection will be used anyway.

Panning the View

Mode: All Modes

Hotkey: Shift MMB / Shift NumPad 2 / Shift NumPad 4 / Shift NumPad 6 / Shift NumPad 8 / Shift Alt LMB

Menu: View View Navigation

Description

To pan the view, hold down Shift and drag MMB in the 3D Viewport. For discrete steps, use the hotkeys Ctrl NumPad 8, Ctrl NumPad 2, Ctrl NumPad 4 and Ctrl NumPad 6 as with rotating. For those without a middle mouse button, you can hold Shift Alt while dragging with LMB.

MMB default behavior

The behavior of the MMB can be customized, in the Preference window, View & Control tab, so it will pan by default and Shift MMB will rotate the view.

Zooming the View

Mode: All Modes

Hotkey: Ctrl MMB / MW / NumPad + / NumPad − / Ctrl Alt LMB

Menu: View View Navigation

Description

You can zoom in and out by holding down Ctrl and dragging MMB. The hotkeys are NumPad + and NumPad −. The View>>Viewport Navigation sub−menu holds these functions too; see (*A 3D Viewport's view menu*).

If you have a wheel mouse, you can perform all of the actions that you would do with NumPad + and NumPad − by rotating the wheel (MW). Since the Buttons window has so many panels, rotating the mouse wheel pans the window left and right in horizontal view. This allows you to pan to the panel you need in a narrow or smaller display, or if the window is narrow. As an alternative, you can easily display the Buttons window vertically; the panels will be arranged top to bottom.

If you have neither a wheel mouse nor a middle mouse button, you can easily zoom in and out with Ctrl Alt LMB.

If You Get Lost...

If you get lost in 3D space, which is not uncommon, two hotkeys will help you: HOME changes the view so that you can see all objects (View>>Frame All Menu entry,) while NumPad . zooms the view to the currently selected objects (View>>Frame Selected Menu entry)

Previous: Manual/Interaction in 3D	Contents	*Next:* Manual/Using the 3D View

 Perspective and Orthographic Projection

User Manual: Contents | Guidelines | Blender Version **2.4x**

Perspective and Orthographic Projection

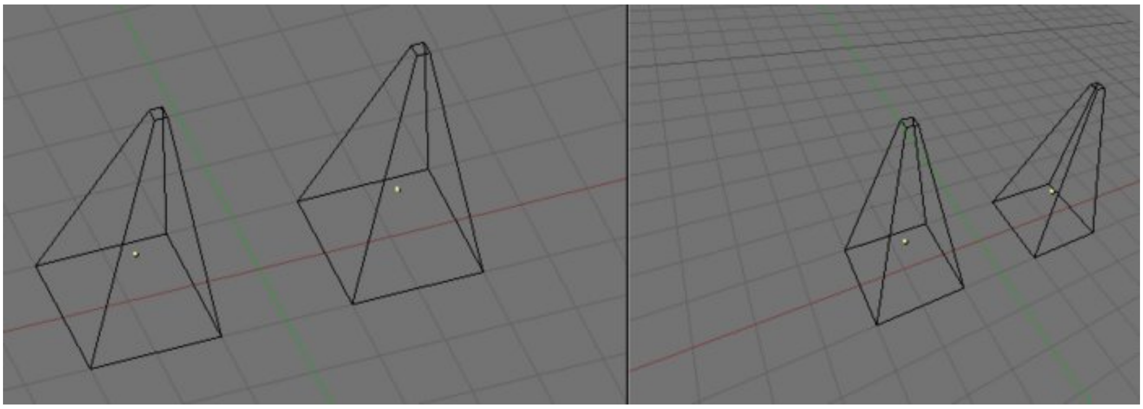
Mode: All Modes

Hotkey: NumPad 5

Menu: View Perspective/Orthographic

Description

Each 3D Viewport supports two different types of projection. These are demonstrated in (*Orthographic (left) and perspective (right) projection.*):



Orthographic (left) and perspective (right) projection.

Our eye is used to perspective viewing because distant objects appear smaller. Orthographic projection often seems a bit odd at first, because objects stay the same size independent of their distance; It is like viewing the scene from an infinitely distant point. Nevertheless, orthographic viewing is very useful (it is the default in Blender and most other 3D applications), because it provides a more "technical" insight into the scene, making it easier to draw and judge proportions.

Options

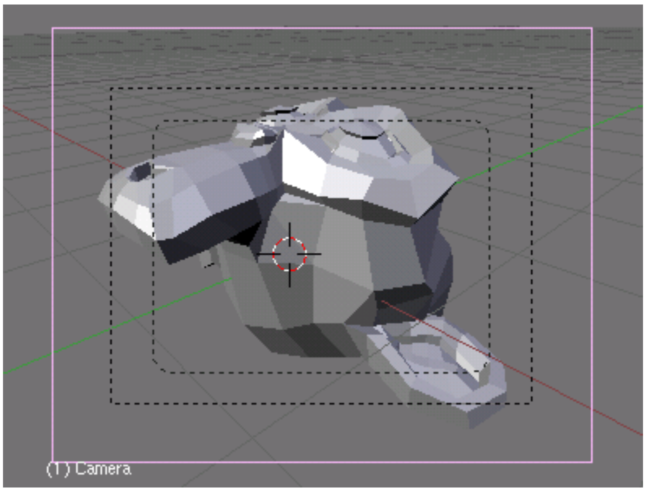
To change the projection for a 3D Viewport, choose the View>>Orthographic or the View>>Perspective Menu entry (*A 3D Viewport's view menu.*). The hotkey NumPad 5 toggles between the two modes.

Camera projection

Changing the projection for a 3D Viewport does not affect the way the scene will be rendered. Rendering is in perspective by default. If you need to create an Orthographic rendering, select the camera and press Ortho in the EditButtons (F9) Camera Panel.

The View>>Camera Menu entry sets the 3D Viewport to camera mode (Hotkey: NumPad 0). The scene is then displayed as it will be rendered later (see *Demonstration of camera view.*): the rendered image will

contain everything within the outer dotted line. Zooming in and out is possible in this view, but to change the viewpoint, you have to move or rotate the camera.



Demonstration of camera view.

Technical Details

Perspective definition

A Perspective view is geometrically constructed this way: you have a scene in 3D and you are an observer placed at a point O. The 2D perspective scene is built by placing a plane, a sheet of paper where the 2D scene is to be drawn in front of point O, perpendicular to the viewing direction. For each point P in the 3D scene a line is drawn, passing from O and P. The intersection point S between this line and the plane is the perspective projection of that point. By projecting all points P of the scene you get a perspective view.

Orthographic definition

In an orthographic projection, also called "orthonormal", you have a viewing direction but not a viewing point O. The line is then drawn through point P so that it is parallel to the viewing direction. The intersections S between the line and the plane is the orthographic projection. And by projecting all point P of the scene you get the orthographic view.

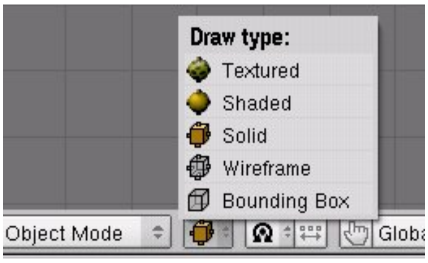
View Shading

Mode: All Modes

Hotkey: Z / Shift Z / Alt Z / D

Description

Depending on the speed of your computer, the complexity of your Scene, and the type of work you are currently doing, you can switch between several drawing modes:



A 3D Viewport's draw mode button.

- Textured Alt Z

  Displays UV image textured models with OpenGL lighting. Procedural textures will not be shown.

- Shaded Shift Z

  Approximates all textures and lighting at each vertex, and blends from one to the next. Much less accurate than using the render engine to check textures, but much faster. Note that if you have no lighting in your scene, everything will remain black.

- Solid Z or Alt Z

  Surfaces are drawn as solid colours, with built−in OpenGL lighting (not dependent on scene light sources)

- Wireframe Z or Shift Z

  Objects only consist of lines that make their shapes recognizable. This is the default drawing mode.

- Bounding Box

  Objects aren't drawn at all; instead, this mode shows only the rectangular boxes that correspond to each object's size and shape.

You can also pop up a contextual menu by pressing D to select between all the draw modes

Local View

Mode: All Modes

Hotkey: NumPad /

Menu: View â†’ Local View

Description

When in local view, only the selected objects are displayed, which can make editing easier in complex scenes. To enter local view, first select the objects you want (see Selecting objects) and then use the View>>Local View Menu entry; use the View>>Global View Menu entry to go back to Global View. (*A 3D Viewport's view menu.*). The hotkey NumPad / toggles between Local and Global View.

View Clipping Border

Mode: Any mode

Hotkey: Alt B

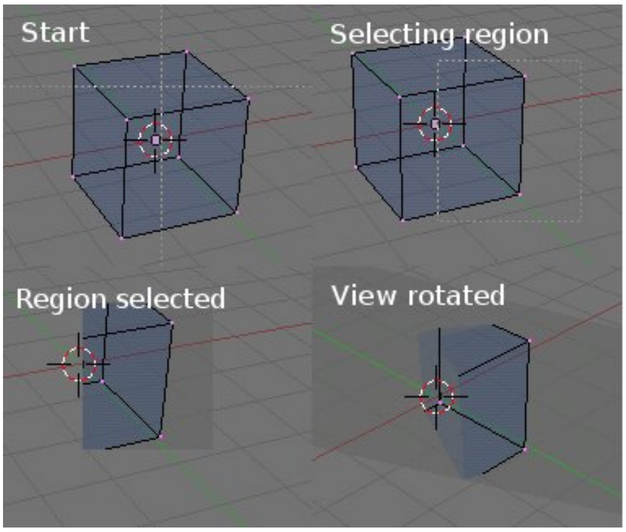
Menu: View Set Clipping Border

Description

To assist in the process of working with complex models and scenes, you can change the view clipping to visually isolate what you're working on.

Using this tool you create a 3D clipping volume shaped liked a Pyramid; you could think of it as a Frustum with a top. You specify the base of the Pyramid by creating a 2D rectangular region.

Examples



Region/Volume clipping.

(*Region/Volume clipping*) is an example of using the clipping tool with a cube. Start by activating the tool with Alt B, see "Start" in the upper left. This will generate a dashed cross−hair cursor. Click with the LMB and drag out a rectangular region shown in the upper right. Now a region is defined and clipping is applied against that region in 3D space. Notice that part of the Cube is now invisible or clipped. Use the MMB to rotate the view and you will see that only what is inside the Pyramid volume is visible. All Edge/Face tools still function as normal but only within the Pyramid volume.

The gray area surrounding the volume is the Pyramid volume itself. To deactivate the clipping tool toggle it by applying Alt B a second time. All of 3D space will become visible once again.

Previous: Manual/Navigating in 3D Space	Contents	*Next:* Manual/3D View Options

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.43**

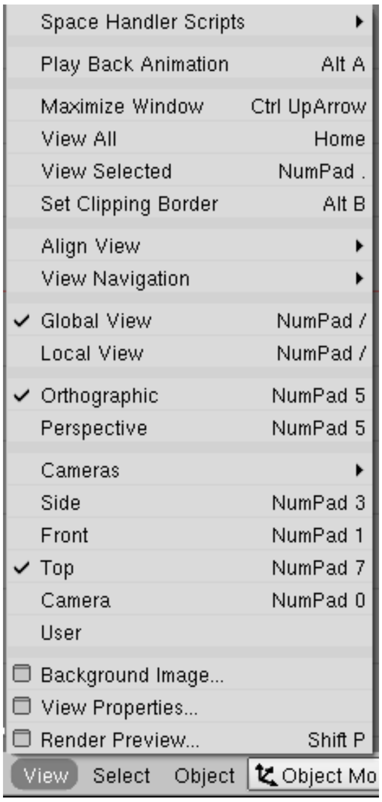
The 3D View is where you perform most of the object modeling and scene creation. Blender has a wide array of tools and options to support you in efficiently working with your mouse, keyboard and keypad. Your flat (two−dimensional) monitor is your viewport into the 3D space.

It is also the oldest, and therefore most feature− and option−rich areas of Blender. DON'T BE INTIMIDATED. Most of us humans never use all the features here, just like we don't use all of our diction on a daily basis either. So, take it slow, a few at a time, experimenting to see what they do.

3D Window Header

The 3D View window is comprised of a workspace and a header. The header is shown at the top or bottom of the workspace, and can be hidden if desired. The header shows you a menu and the current mode, as explained below.

View Options



This menu provides options to control the way the workspace is viewed:

- **Space Handler Scripts** − This submenu shows Space Handler Scripts; by default, there aren't any.
- **Play Back Animation** − This item plays back the animation from the current frame.
- **Maximize Window** − This item maximizes the 3D View Window to fill the entire Blender window, and once selected this menu item will change to Tile Window, if menu entry Tile Window is then selected the 3D View Window will be restored to it's previous size. Using the menu entry to Maximize/Tile has the affect of limiting it's affect to just the 3D View Window. As well as the menu entry, the shortcuts Ctrl and Ctrl can also act as toggles to Minimize/Tile not only the 3D View Window but also any window which currently has focus. In addition to the shortcut Ctrl and Ctrl , it is also possible to Maximize/Tile the currently focused window with shortcut Shift Space, making it extremely convenient for laptop users, as they can quickly Maximize/Tile the currently focused window to work on another window such as the Buttons Window/Outliner Window for example.
- **View All** − This command zooms the 3D view to encompass all the objects in the current scene.
- **View Selected** − This command zooms the 3D view to encompass all the selected objects.
- **Set Clipping Border** − This command allows you to define a clipping border to limit the 3D view display to a portion of 3D space. For more information on this command, see View Clipping Border in the Using the 3D View section of the manual.
- **Align View**−This submenu (shortcut C) shifts your view to be centered on the cursor. Shift−Center (shortcut Shift C) centers your view **and** zooms out so that you can see everything in your scene. You can also change your viewpoint to be through the camera, and center the camera on the cursor. Instead of the cursor, you can center your view on the selected object (shortcut keypad *)
- **View Navigation** − This submenu contains commands for rotating and panning the view. Using these commands through the menu is not that efficient; however, like all Blender menus, the much−more−convenient keyboard shortcuts are listed next to the commands. Fly mode moves your view through 3D space. Use the keys indicated to orbit your view, or hold down MMB and move your mouse. Use the keypad NumPad +/NumPad − keys to zoom, or scroll your mousewheel.
- **Global /Local View** − Global view shows all of the 3D objects in the scene. Local view only displays the selected objects. To switch between global and local view use NumPad /. Accidentaly pressing NumPad / can happen rather often if you're new to Blender, so if a bunch of the objects in your scene seem to have mysteriously vanished, try turning off local view.
- **Orthographic**, **Perspective** − These commands change the projection of the 3D view. For more information, see Perspective and Orthographic Projection. Generally you want to stay in Orthographic view.
- **Cameras** − This submenu lists all the cameras in the scene. Selecting one will make it the active camera; there is also a command that sets the current object (which doesn't have to be a camera) as the camera, so you can see what the scene looks like from its point of view.
- **Side**, **Front**, **Top** − These commands change the view to the default side, front, or top views. Pressing the Ctrl key changes to the 'other' corresponding view: Ctrl−NumPad 3 right side, Ctrl−NumPad 1 back, or Ctrl−NumPad 7 bottom−looking−up views.
- **Camera** − This command switches the view to the current camera view.

- **User** − This command switches to a user view. In most cases, this won't seem to do anything, but if you are in the camera view or have orthographic projection on, the view will change to perspective (and leave the camera view, if applicable).

- **Background Image...** − This command will toggle the Background Image floating panel, which allows you to load and pick an image to display in the background of the orthographic 3D view, as well as adjust its size and position. This is useful if you have a picture (for example, a face) that you want to model from. Each pane (3D window view) has its own background image settings. Each pane can or cannot use background image independently. So, you can set top view to have one image, but unless you set the others to use an image, no other views will use it. Side view can have another, and front another. They can all be the same image if the image is one big composite of all views you want to reference; just use the offset values in each pane to position the image where you want it. Background images can be stills, movies (avi or sequences) or even a render from another scene. For movies, enable Auto Refresh and Blender will display the appropriate frame from the movie when you change frames in your animation.

Use Lo−Res Proxy:

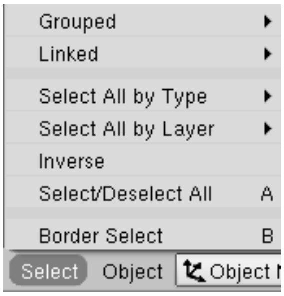
To improve PC performance when using background images you may have to use lower−resolution proxies. If your monitor resolution is 800Ã—600, then the background image, full screen, without zooming, only needs to be 800Ã—600. If your reference image is 2k x 2k, then your computer is grinding away throwing away pixels. Try instead to take that 2kx2k image, and scale it down (using Blender, or Gimp) to, for example, 512Ã—512. You will have 4x the performance, with no appreciable loss of quality or exactness. Then, as you refine your model, you can increase the resolution.

- **View Properties...** − This command toggles the View Properties floating panel, which allows you to toggle the grid and adjust its spacing, adjust the zoom of the camera, toggle specific axes (X, Y, and Z), view and change the specific location of the 3D cursor, adjust several toggles (outlining the selected object, showing all object centers, showing relationship lines), and, lock the 3D view so that it always points towards a certain object or bone. See View Properties Window for more details.
- **Render Preview...** − This command toggles the Render Preview panel, which shows a (relatively) live preview render of whatever it is over.

- **Onion Skinning/Ghosting** − Selecting Keyframe display mode (pressing K) in the IPO Window, and possibly then pressing K again in the 3D View (depending on your PC's OpenGL support) will display the keyframed locations for an object (mesh, armature, etc.) Showing the ghost past/present keyed postions IPO Keys is an often overlooked feature that can greatly assist animation visualization. The location of the object at the current frame is shown as a green line in the IPO Window, and as the object in the 3D View. The keyframe selected in the IPO Window is shown in yellow, as is the outline of the object in the 3D View, further helping you visualize the animation. All other ghost keyframed locations are shown as a black outline in the 3D View. Ghosting for armatures is more versatile and thus more complicated; see the Armatures section of the user manual for Armature Visualization options.

In addition, the NumPad * key orients the view normal to the selected face and enter homes the display to fit everything within the view.

Select Menu



- **Grouped** − Blender has a few ways of grouping items together. This submenu contains commands to select objects by their various groupings.
  - **Children** − This command selects all the children of the selected object(s) (as in their children, the children of their children, etc.)
  - **Immediate Children** − This command selects the children of the select object(s); however, unlike the previous command, it does not continue selecting the children of the children, just the direct descendants of the parent.
  - **Parent** − This command selects the parent(s) of the selected object(s).
  - **Siblings (Shared Parent)** − This command selects all the objects that shared the parent of an object. This means that if you have, for example, several Blender objects that make up one physical object that are all children of one part of the object of an empty, you can select one Blender object that is part of the physical object, use this command, and all the Blender objects that make up the physical object will be selected (if that made any sense!).
  - **Objects of Same Type** − This command selects all objects of the same type (Lamp, Mesh, Camera, etc.)
  - **Objects on Shared Layers** − This command selects all the objects on the same layers as the selected object(s). {I think.}
  - **Objects in Same Group** − This command selects all the objects in the groups that the selected object(s) are in. This can be used for the same purpose as **Siblings (Shared Parent)** if you have grouped parts of a physical object instead of using parents.
  - **Objects Hooks** − This command selects any objects that are acting as Hooks for the selected object(s).
- **Linked** − This submenu contains commands that allow you to select objects based on data (Ipo curves, Materials, Textures, etc.) that they share.
  - **Object Ipo** − This command selects all the objects that share the Ipo curves of the selected object(s).
  - **ObData** − This command selects all the objects that share the ObData of the selected object(s).
  - **Material** − This command selects all the object(s) that share the Material(s) of the selected object(s).
  - **Texture** − This command selects all the object(s) that share the Texture(s) of the selected object(s).

- **Select All By Type** − This submenu contains commands that allow you to select all the objects of a certain type (Mesh, Camera, Lamp, etc.)
- **Select All By Layer** − This submenu contains commands that allow you to select all the objects in a specified layer.
- **Inverse** − This command inverts the selection (selects all the deselected objects and deselects all the selected objects.
- **Select/Deselect All** − This command deselects the current selection if there is one; if nothing is selected, it selects everything.

- **Border Select** − This command allows you to select objects using the traditional rectangle that most programs use. After pressing B or selecting this menu option, click LMB and drag your mouse diagonally through your workspace. When you release your LMB, all objects that were totally

within the box will be selected. If you instead click MMB or RMB and drag, all objects within the box will be *de*−selected.

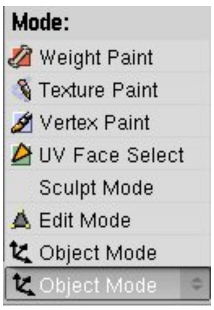
Object Menu

This menu operates on objects as a whole. Many options act on the active object, based on other objects. You indicate that the Blender by RMB clicking on the base object, then Shift RMB clicking on the active object, and then invoking the menu option. In the case of wanting to work on more than two objects, simply click on all the base objects first; the last object selected will be the active object.

- **Scripts**: this submenu lists available python scripts written to extend blender's object handling capability. Of note is the Knife tool that cuts meshes. See each script's documentation for more information on how to use the script.
- **Move to Layer**: Objects can be organized into 20 Layers, and only a few layers can be selected for display to avoid clutter. This option moves the selected object(s) to a layer. To do so, select one of the 20 layers by clicking on one of the buttons in the popup window and click the OK button. If the layer is not selected for display, the object is removed from view. To view the objects on a layer, click the appropriate Layer button located to the right on the header.
- **Convert Object Type**: Objects can have Modifiers which can be turned on or off; this option applies those modifiers permanently.
- **Join Objects**: joins multiple selected objects into on single object.
- **Boolean Operation**: This submenu allows you to perform discrete operations on the active object based on previously selected objects. Union extends the object to include all selected objects. Difference boolean modifies the active object by cutting away parts where they intersect. Intersect discards everything except where they intersect. The first operations are destructive; they can also be made temporary via the Modifier option.
- **Constraints**: You can constrain an object based on another object, like a dog on a leash. Various ways to constrain the active object (based on another previously selected object) are by location, rotation, scale, etc.

- **Track**: You can make an object face, or always point to another object by the Track option. If you move the base object, the active object will rotate so that it always "keeps an eye" on the base object.
- **Group**: Grouping is a completely arbitrary way for you to group like things together. If you are making a scene of a park, you can group all the trees together if you like.
- **Parent**: Use this menu to designate the active object as the parent of one or more child objects. When you then move the parent, the children move with it.
- **Copy Attributes**: An object is in a certain place, called a location. "Location" is an attribute of the object. All selected objects can copy the attributes of active object using this option.
- **Make Local**: If an object was linked from another scene, this makes a local copy of it for the current scene.
- **Make Single User**: An object, like an 8−ball, can share the same mesh as another object. In this case the Mesh is called "billiards ball" and has 16 users (15 balls plus the cue ball). Making a local copy assigns a copy of the multi−user mesh to the selected object. You can then edit the local copy without affecting any of the other users.
- **Make Links**: To Scene This is a way to proxy an object to another scene. It exists in the current scene, but will show up in the linked scene as well. To Object Ipo, Mesh data, etc is a way to make the current object join the multi−user community and share the selected item.
- **Delete**: Wipes it out, man. Like totally. If its material and texture were single user and now are not used, they will not be saved with the .blend.
- **Duplicate Linked**: Makes a copy of the object, but links their mesh as multi−user, so if you change one (like a table leg), all the other table legs match.
- **Duplicate**: Makes a xerox copy. Well, a three−dimensional xerox copy, if there is such a thing.
- **Insert Keyframe**: Records the current location, rotation, etc (whatever you select) of the object at that frame. Use this to set up basic animation.
- **Snap**: This menu allow you to move the select object to the cursor, grid or vice versa. Very handy in making items share the same space
- **Clear/Apply**: Clears (resets to zero) the object's scale/rotation as selected, or applies current rotation/scale to the object making them default.
- **Mirror**: see previous section, half a monkey example.
- **Transform**: Use the menu to refresh your memory of the most common hotkeys.
- **Transform Properties**: Pressing N pops up a floating panel that gives you key information about the active object: location, rotation, scale, dimension.

Mode List

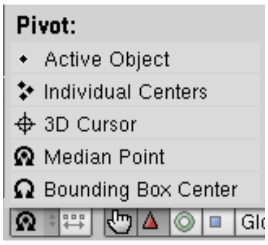


Blender has a few modes of operation; working on **objects** as a whole, or in **edit** mode by allowing you to modify the shape of the object. In **Sculpt** mode, your cursor becomes a tool to shape the object, while your cursor becomes a brush in Vertex Paint, Weight Paint, or Texture Paint modes.

ViewPort Shading List

see Manual/Using the 3D View

Rotation Selector



When rotating or scaling an object or bunch of vertices/edges/faces, you may want to shift the pivot point in 3D space. Using this selector, you can change the pivot point to the location of the cursor, the average center spot of the selected items (median), etc. Pivoting is fully described here.

Use the Object Center toggle to switch between multipoint averages or last selected as the center.

Transform (Manipulator) Selectors

These handy selectors, also featured in other not−to−be−named CG packages, allow you to rotate or move objects by grabbing (clicking with your mouse) their controls and moving your mouse in the axis. Each color stands for an axis.

Click on the little finger icon to enable the manipulator. Your object in 3D View will now have the manipulator around its center. The **transform** manipulator is a 3−axis arrow; the **rotate** manipulator are three circles, one for each axis; click and drag on the arc to rotate on an axis. The **scale** manipulator is 3−axis of lines that end in a block. Customize the size of the manipulators in the User Preferences.

LMB click the buttons for a 3D Move, Rotate, or Scale Selector. Shift LMB to activate multiple manipulators at the same time. You can move, rotate or scale according to a Global view, or local, view, etc. Generally, stick with Global view until you get the hang of things.

Layer Selector

Layers are pretty well documented here. In particular, selecting layers to see in is covered in that section on Viewing layers and Moving objects between layers is also discussed on a separate Layers page.

Render Button

The Render button renders an OpenGL copy of the 3D view. It's pretty much exactly what you see minus the grid, axes, etc; it just shows the objects. It uses the same Drawmode as the 3D view, so it's rather useful if someone asks to see the wireframe of an object you're working on.

Ctrl LMB or Shift LMB clicking the button will render an animation of the 3D View, making it useful for making preview renders of animations. The animation will be saved in the rendered Pics folder (Scene (F10) context, Output panel, top filespec) in the format as an avi file or image sequence, depending on the format you have chosen (Scene (F10) context, Format panel), for the number of frames specified in the Sta: and End: fields (Scene (F10) context, Anim panel).

Using the 3D Window

Description

Your window pane is just like a window looking in on a 3D world. To help keep you oriented as to which way is up (Z), an XYZ axis orientation indicator is in the bottom left hand corner, along with the number of the frame you're working on and the name of the active object. The rest of the view, if in one of the normal orthogonal views (front, side or top) will show a grid. Each line in the grid is a blender unit (BU). A BU is an arbitrary unit of measurement. If you are modeling something from the real world, you can set (in your mind) a BU equal to whatever unit of measure your country and culture favor at the moment. If you are a Swiss CERN physisist, then perhaps Angstroms are your thing. If you are a German Engineer, then Millimeter might be in order. If you are an Amsterdam Space Cadet, then Astronomical Units might light up your fancy.

In this 3D space, the active object is hightlighted in pink. Your cursor is a red−white circle with a scope crosshairs. LMB clicking moves the cursor. Use the Snap button to move the cursor by some means other than aimlessly clicking around.

RMB clicking selects the object being pointed to unless it is already selected, in which case RMB clicking de−selects it, like a toggle. Shift RMB button selects another object and keeps the first one(s) still selected, allowing you to select multiple objects (remember that the last one selected is the active one).

Editing Mode

Enter Edit mode by the mode selector, or by pressing Tab in the window. In edit mode, you select the components of the object, and do things to them. Strange, horrible things. Unless you're good at this stuff and are willing to put in a lot of practice, in which case you'll get better. Some objects, like cameras, cannot be edited. Press Tab again to return to the mode from whence you came.

3D Window Toolbox (Popup Menu)

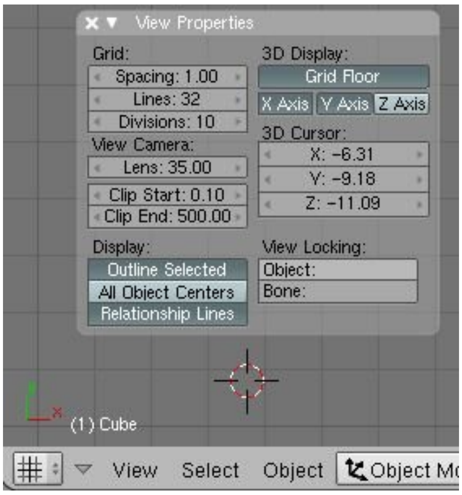
Pressing space in the 3D window pops up a very handy little menu. You can also press LMB or RMB for one second to do so.

Add

Use this option to add objects to your scene. You can get to this option by

- clicking the Add menu item in the User Preferences header at the top of your screen, or
- pressing space when your mouse cursor is hovering over any 3D view window

The object, when it is added, is placed wherever your 3D cursor is, and the object is automatically placed in Edit mode because Blender assumes you want to start modifying it right away. To set the location of your 3D cursor, you can:



- LMB click in a 3D View window, and the cursor jumps to that spot. To place the cursor in 3D space, you will have to click in two windows that have different perspectives; once for example in the top view to establish the XY location, and then again in a front view to establish the Z location,
- Use the View −>Properties window and enter an EXACT XYZ location in the 3D Cursor fields, or
- Select an object whose center is where you want the 3D cursor to be, and Shift Snap the Cursor −> Selection, and the cursor will jump to the selected object's location. Note that an object added there will be put inside or will have its surface mushed with the other objects (the two objects will attempt to occupy the same space).

Blender supports many different primitives which can be added to your scene. If you add an object in Object mode, the primitive is added as a separate object to the scene. If you add an object while in Edit mode for another object, the primitive is added to the the other object, forming a compound object from many primitives:

- **Mesh** − this submenu allows you to add polygonal meshes to your scene. The monkey (Suzanne) is quite useful for testing purposes (trying out new materials, fur, etc.)
- **Curve** − this submenu allows you to add curves or paths to your scene. These are useful for modeling curved objects (roller coasters, legs of fancy furniture, etc) or for making curves for animated objects to follow.
- **Surface** − this submenu allows you to add NURBS objects to your scene.
- **Meta** − this submenu allows you to add meta objects to your scene. These are algorithmically−generated objects, and are quite useful for special effects (using two metaballs to animate a cell splitting in half, for example).
- **Text** − this command allows you to add Text objects to your scene.
- **Empty** − this command allows you to add Empties to your scene. Empties aren't rendered; rather, they are often used to control aspects of other objects. For example, you could use an Empty to control the rotation of an array modifier. Since they don't render, you can use them for all sorts of things such as hooks, etc. that require an object to get a location, rotation and/or size from.
- **Group** − this submenu allows you to add copies of any groups you have in your scene. This is quite useful if you have entire objects grouped together, as you can easily add copies of them.
- **Camera** − this command adds a camera to your scene.
- **Lamp** − this submenu contains various types of lamps you can add to your scene. For more information on the different types of lamps, see Lamp Types in the manual.
- **Armature** − this command adds an Armature (skeleton) to your scene. These are primarily used for animating arms, legs, etc, though if you are in a Pixar−y mood, you can always rig up a lamp!
- **Lattice** − this command adds a Lattice to your scene. These objects do not do anything themselves; however, you can use them to deform objects. In order to have a Lattice deform an object, you have to add a Lattice Modifier to the object you want to deform. You can then place the Lattice around the object (like a cage), and any changes to the Lattice will deform the object. The more detailed the object you are deforming is, the better it will look, so a Subsurf modifier may be helpful here.

Edit

- **Enter Editmode** − this command will enter Edit mode, which allows you to edit the vertices, edges, and faces of a specific object, rather than manipulating the entire object.
- **Duplicate** − this command makes a separate duplicate of the selected object(s).
- **Duplicate Linked** − this command makes a duplicate of the selected object; however, the ObData is shared, so the objects share the same mesh (if applicable), Ipo curve, etc.
- **Delete** − this command deletes the selected object(s).
- **Object Keys** − this submenu contains commands related to keyframes.
    - **Show/Hide** − this command toggles showing wireframe version of the keyframes for the selected object. This is quite useful, as it allows you to visualize the path of the selected object.
    - **Select Next** − this command selects the next keyframe for the selected object.
    - **Select Prev** − this command selects the previous keyframe for the selected object.

Select

- **Grouped** − This submenu contains commands to select objects by various groupings.
    - **Children** − This command selects all the children of the selected object(s) (as in their children, the children of their children, etc.)
    - **Immediate Children** − This command selects the children of the select object(s); however, unlike the previous command, it does not continue selecting the children of the childrden.
    - **Parent** − This command selects the parent(s) of the selected object(s).
    - **Siblings (Shared Parent)** − This command selects all the objects that shared the parent of an object. This means that if you have, for example, several Blender objects that make up one physical object that are all children of one part of the object of an empty, you can select one Blender object that is part of the physical object, use this command, and all the Blender objects that make up the physical object will be selected (if that made any sense!).
    - **Objects of Same Type** − This command selects all objects of the same type (Lamp, Mesh, Camera, etc.)
    - **Objects on Shared Layers** − This command selects all the objects on the same layers as the selected object(s). {I think.}
    - **Objects in Same Group** − This command selects all the objects in the groups that the selected object(s) are in. This can be used for the same purpose as **Siblings (Shared Parent)** if you have grouped parts of a physical object instead of using parents.
    - **Objects Hooks** − This command selects all the objects that are acting as Hooks for the selected object(s).
- **Linked** − This submenu contains commands that allow you to select objects based on data (Ipo curves, Materials, Textures, etc.) that they share.
    - **Object Ipo** − This command selects all the objects that share the Ipo curves of the selected object(s).
    - **ObData** − This command selects all the objects that share the ObData of the selected object(s).
    - **Material** − This command selects all the object(s) that share the Material(s) of the selected object(s).
    - **Texture** − This command selects all the object(s) that share the Texture(s) of the selected object(s).

- **Select All By Type** − This submenu contains commands that allow you to select all the objects of a certain type (Mesh, Camera, Lamp, etc.)
- **Select All By Layer** − This submenu contains commands that allow you to select all the objects in a specified layer.

- **Inverse** − This command inverts the selection (selects all the deselected objects and deselects all the selected objects.
- **Select/Deselect All** − This command deselects the current selection if there is one; if nothing is selected, it selects everything.

- **Border Select** − This command allows you to select objects using the traditional rectangle marquee (which isn't actually display as a marquee) that most programs use.

Transform

- **Grab/Move** − This command allows you to freely grab (move/translate) the selected object(s).
- **Grab/Move on Axis** − This submenu contains commands that allow you to move an object along a specific axis: **X Global**, **Y Global**, etc.

- **Rotate** − This command allows you to rotate an object about the view Z axis (i.e. it turns clockwise/counterclockwise around the screen; the rotation axis goes straight into the display.)
- **Rotate on Axis** − This submenu allows you to rotate an object about a specific axis. **X Global**, **Y Global**, etc.

- **Scale** − This command scales the selected object(s).
- **Scale on Axis** − This submenu contains commands that allow you to scale an object on a certain axis: X Global, etc.

- **ObData to Center** − This command moves the Mesh/Curve Points/etc. of an object so that they are centered according to the current object center.
- **Center New** − This command moves the center of the object to the center of the object data.
- **Center Cursor** − This command moves the center of the object to the current location of the 3D cursor.

- **Properties** − This command toggles the Transform Properties floating panel, which allows you to input exact locations for vertices, as well as location, rotation, and size for entire objects.

- **Mirror** − This command allows you to mirror (flip) the selection about the appropriate axis. As with all axis−related stuff, Global refers to the view in general, while Local refers to the axis specific to that object.

- **Snap** − This submenu contains commands that allow you to snap the 3D cursor and the selection to the grid and each other.
    - **Selection −> Grid** − This command snaps the selection to the nearest point on the grid.
    - **Selection −> Cursor** − This command snaps the selection to the location of the 3D cursor.
    - **Cursor −> Grid** − This command snaps the 3D cursor to the nearest point on the grid. This is quite useful if you manually clicked to position the 3D cursor, and it didn't land exactly where you wanted.
    - **Cursor −> Selection** − This command snaps the 3D cursor to the center of the selected object(s).
    - **Selection −> Center** − This command snaps the selected object(s) to the center of the selected object(s). This is most useful in edit mode, as it allows you to snap vertices, edges, or faces to the center of the object you're working on.

- **Clear/Apply** − This submenu contains commands that allow you to clear (reset) or apply (make permanent) the location, rotation, scale, deformation, or duplicates of the selected objects.
    - **Clear Location** − This command clears (resets) the location of the selected object(s) to 0,0,0.
    - **Clear Rotation** − This command clears (resets) the rotation of the selected object(s).

- ♦ **Clear Scale** − This command clears (resets) the scale of the selected object(s).
- ♦ **Apply Scale/Rotation** − This command applies the scale and rotation. The object data (mesh/curve points/etc.) is modified so that the scale is 1 and the object isn't rotated at all.
- ♦ **Apply Deformation**
- ♦ **Make Duplicates Real** − This command makes the duplicates (from using DupliVerts or DupliFrames) real objects (so you can edit them individually).

Render

- **Passepartout** − This option toggles the Passepartout option of the selected camera. When turned on, it darkens the area around the camera, allowing you to focus on the area that's actually going to be rendered.
- **Set Border** − This option allows you to drag to select a specific area of the camera view to render. This is useful if you're tweaking a specific detail of an object and don't want to render the entire scene (If you're tweaking an entire object, Local View may be more of what you're looking for). If you want to remove this clipping region from your future rendering, uncheck the Border button (checked by default when you use the Set Border option) in the bottom of the Render panel in the Scene (F10) context and Render buttons subcontext.
- **Render** − This option renders the current scene. It's the same as the big RENDER button in the button window, and can also be activated with F12.
- **Anim** − This option renders an animation using the current animation settings. It's the same as the ANIM button in the button window, and can be activated by Ctrl F12.
- **Preview** − This toggles the Preview Render floating panel, which displays a preview (non−antialiased) version of whatever portion of the 3D view is currently underneath it.

Previous: Manual/Using the 3D View	Contents	*Next:* Manual/Camera View

User Manual: Contents | Guidelines | Blender Version **2.43**

Add a new camera

Mode: Object Mode

Hotkey: Shift A to add new, F9 to change settings.

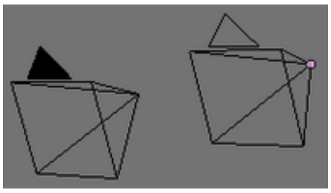
Menu: Add â†’ Camera

In object−mode simply press space and in the popup menu, choose Add−−>Camera. New cameras are directed in parallel with the current viewport.

Change active camera

Mode: Object Mode

Hotkey: Ctrl NUM0



Active Camera

Active camera is the camera that is currently used for rendering. Select the camera you would like to make active and press Ctrl NUM0 (by doing so, you also switch the view to camera view). In order to render, each scene **must** have a camera.

The active camera is the one with the filled *Up* triangle on top seen in the 3D viewport. The left camera in the picture.

The Active Camera for rendering purposes is tied to the "Lock Layers and Used Camera to Scene" button (lock icon) for each 3D view port. Whichever camera was designated Active in a viewport where this option is activated (locked) becomes the rendering camera. If all 3D views are unlocked (as mine were to display the various views simultaneously), no matter which camera is made "Active" for the unlocked viewport, only the camera from the previously locked view will be the render camera.

Move active camera to view

Mode: Object Mode

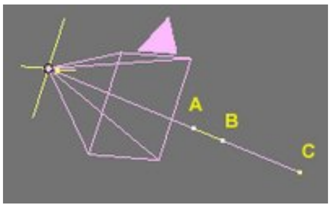
Hotkey: Alt Ctrl NUM0

Moves the selected camera to current 3D view. Select a camera and then move around in 3D view to a desired position and direction for your camera. Now press Alt Ctrl NUM0 and your selected camera positions itself at

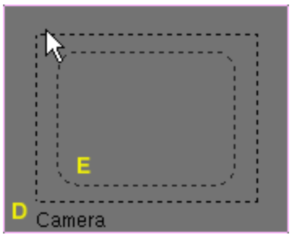
your spot and switches to camera view.

Camera Settings

Camera panel



Camera



Camera View

Cameras are invisible in a scene; they are never rendered, so they don't have any material or texture settings. However, they do have Object and Editing setting panels available which are displayed when a camera is the active (selected) object.

- Lens − Represents the lens in mm. If Orthographic is selected, this will change to a Scale variable.
- DoFDist − Distance to the point of focus. It is shown as a yellow cross on the camera line of sight. Limits must be enabled to see the cross. It is used in combination with the Defocus Node

- Orthographic − Toggles Orthographic mode for rendering. See Manual/Using the 3D View for a more detailed description on Orthographic mode. You can limit what is rendered by moving the camera. Objects behind the camera's XY plane are not rendered. When enabled, the Lens Size field changes to a Size field, which includes more/less area in the render.
- Clipping Start/End − Sets the clipping limits. Only objects within the limits are rendered. If Limits is enabled, the clipping will be seen as two yellow dots on the camera line of sight. **C** on the Camera picture. The first is at Camera origin.
- Shift X/Y − Shifts the camera viewport.
- Limits − Toggles viewing of the limits on and off.

- Mist − Toggles viewing of the mist limits on and off. The limits are shown as two white dots on the camera line of sight. **A** and **B** on the Camera picture. The mist limits are set in the World Panel F8.
- Name − Toggle name on and off. **D** on the Camera View picture.
- Title Safe − When title safe is enabled an extra dotted frame is drawn inside the camera viewport. Shown beside **E**.
- Passepartout Alpha − This mode draws the area outside of the camera's field of view with a different darkness, set by the Alpha setting.
- Size − The draw size of the camera in the 3D view. This doesn't affect the camera's output, it is just a convenience to enable easier selection of the camera object in the viewport. (The camera object can also be scaled using the standard S transform key).

Camera Navigation

To control the camera while viewing through it NumPad 0:

Aiming the camera

Press Shift F to enter "Camera Fly Mode", then move the mouse around to aim the camera, LMB to set the new orientation, RMB or ESC to cancel.

Rolling

To roll the camera, the camera needs to be selected (while viewing throught it, RMB on the solid rectangular edges selects it), then press R to enter standard object rotation mode, the default will be to rotate the camera in it's local Z axis.

Pitch

To rotate along the local X axis, press R, then XX. The first X (or Y or Z for other axis) selects the global axis, pressing the "axis letter" a second time selects the local axis (this works when rotating any object).

Dolly

To dolly the camera, press G then MMB , LMB to complete.

Track Camera

Press G(rab) and move the mouse (LMB to set position).

Previous: Manual/3D View Options	Contents	*Next:* Manual/Layers

User Manual: Contents | Guidelines | Blender Version **2.43**

Layers

Mode: Object Mode

Panel: Object Draw

Hotkey: M

Menu: Object Move to Layer...

Description

3D scenes often become exponentially more confusing with growing complexity. Also, sometimes the artist needs precise control over how individual objects are lit, and do not want lights for one object to affect nearby objects. For this and other reasons below, objects are placed into one or more "layers". Using Object Layers, you can:

- By selecting certain layers in the 3D−View header bar, only objects on those layers are displayed at any one time in your 3D View, speeding up refresh/redraw, reducing virtual−world clutter, and enhancing workflow velocity.
- By making a light illuminate only those objects on its layer, then you control which lights illuminate an object, and vice versa
- Since Particles are affected by forces and effects on the same layer, you can control which forces effect which particle systems
- Renderlayers cause the rendering of objects on certain layers. By using Renderlayers, you control which objects on currently selected layers are rendered, and which properties/channels are made available for compositing.

Armatures can become very complex, with different types of bones, controllers, solvers, custom shapes, and so on, all withing a fairly close space, can also become cluttered. Therefore, Blender provides layers just for Armatures. Armature layers are very similar to object layers, in that you can divide up an armature (rig) across layers and only display those layers you wish to work on. Layers (armature and object) behave the same way, Armature layers are discussed in the Armatures section.

3D layers differ from the layers you may know from 2D graphics applications: they have no influence on the drawing order and are there (except for the special functions listed above) mainly to provide the modeler with a better overview.

When rendering, Blender only renders those layers selected. If all your lights are on a layer that is *not selected*, you won't see anything in your render except for objects lit by ambient lighting.

Groups and Parenting are other ways to logically group related sets of objects. Please refer to those applicable sections for more info.

Options

Viewing layers

Blender provides 20 layers; you can choose which are to be displayed with the small unlabeled buttons in the header (*A 3D Viewport's layer buttons.*). To select only one layer, click the appropriate button with LMB ; to select more than one, hold Shift while clicking.

A 3D Viewport's layer buttons.

To select layers via the keyboard, press 1 to 0 (on the main area of the keyboard) for layers 1 through 10 (the top row of buttons), and Alt 1 to Alt 0 for layers 11 through 20 (the bottom row). The Shift key for multiple selection works for these hotkeys too. By default, the lock button directly to the right of the layer buttons is pressed; this means that changes to the viewed layers affect all 3D Viewports. To select only certain layers in one window, deselect locking first.

Multiple Layers

An object can exist on multiple layers. For example, a lamp that only lights objects on a shared layer could "be" on layers 1, 2, and 3. An object on layers 3 and 4 would be lit, whereas an object on layers 4 and 5 would not. There are many places were layer−specific effects come into play, especially lights and particles. To place an object on multiple layers, Press M and then shift−click the layers you want it to be on.

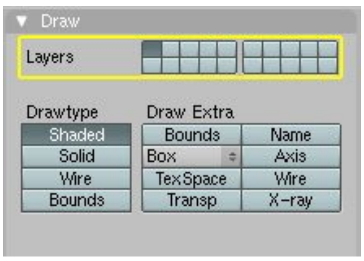
Moving objects between layers

Layer selection

To move selected objects to a different layer, press M, select the layer you want from the pop−up dialog, then press the Ok button. Objects can be on more then one layer at a time. To move an object to multiple layers, hold Shift while clicking. If you wish to clone−display the object to additional layers, be sure to Shift LMB click the original layer as well.

Object pane selection

Another way to view or change a selected object layer is via the draw pane, this can be located by pressing F7 or clicking on the panel object icon as shown



Object draw pane layers

You will then see the layer buttons in the draw pane, as before the object can be displayed on more then one layer by hold Shift while clicking. In this way, you can control what objects are shown on each layer. Layers are useful in keeping your display from being too cluttered, from refresh/redraw time being too long, and to give you control over what to render for later compositing.

Animating Layers

An Object's layer "membership" can be animated E.g. to have objects suddenly appear or disappear in a scene.

Example of Object layer arrangement

As a suggestion, use the top row of layers for the real important things, and the bottom row for those you don't use or change often, or for alternatives for the top row. In a staged set involving mainly two actors, then, you might have, for Layers:

1. Lead Actor
2. Supporting Actor
3. Supporting Crew (background actors)
4. Particles and effects (vortex, wind)
6. Main Stage
7. Main backdrops and panels
8. Main props (tables, chairs)
9. Little props, fillers, decorations, trappings
10. Cameras, Lights

11. Lead Actor's armature
12. Supporting Actor's armature
13: Crew armatures
14: alternative clothing
15: mesh WIP
16. different stage setup, dimensions
17. different backdrops that maybe we should use
18. other big props that maybe clog up the scene
19. props WIP
20. Additional lighting

Layer Naming Script

Layer Manager Scripts:

There are also a few scripts available that allows you to give names to layers.

Links:

- 4mm Layer Manager Script
- Layer Manager Script

Previous: Manual/Camera View | Contents | *Next:* Manual/LocalGlobal

User Manual: Contents | Guidelines | Blender Version **2.43**

Local or Global View

Mode: Object Mode or Edit Mode

Hotkey: NumPad /

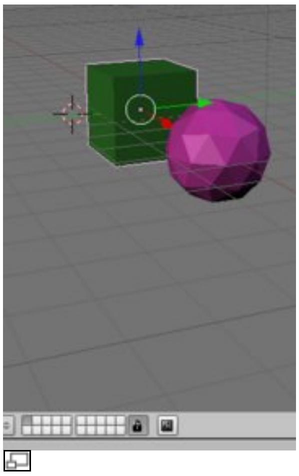
Menu: View â†’ Local View or Global View

Description

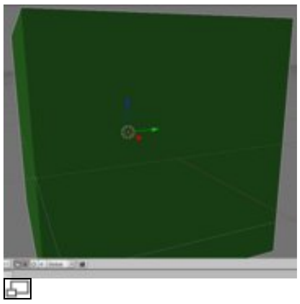
Toggles between Local and Global view mode. The currently selected object is the focus for the mode. An object must be selected to enter Local view mode.

Examples

The Layers on the 3D view header disappears while in Local view mode.



Global View.



Local View.

In (*Global View*), the Layer buttons are visible and the green cube is selected. (*Local View*) shows the cube has been focused and centered in the 3D view and the Layer buttons have disappeared. This feature is handy when you want to focus on just the object and nothing else. If a scene has thousands of objects visible, it can potentially speed interactivity up because it is the only object visible.

Previous: Manual/Layers Contents *Next:* Manual/Manipulation in 3D Space

User Manual: Contents | Guidelines | Blender Version **2.48.1**

What is Grease Pencil?

This page is based on the grease pencil release logs for blender 2.48

The ability to draw in and/or on viewports using freehand strokes to form sketches and annotations has many benefits for collaborative communication and planning. This can be linked back to traditional 2D−workflows with pencil and paper, where rough 'guideline' sketches were often used for planning and also for communicating ideas quickly. It is often useful to be able to directly scribble on to a work in progress, instead of having to do so in a separate place (i.e. another part of the window, or even in a different application altogether).

The name "Grease Pencil" is derived in homage to the wax crayons/pencils that early CG Animators used to draw arcs and other planning notes on their CRT's with.

In addition to uses for animators in planning their poses and motion curves, Grease Pencil can also be useful in a number of scenarios, including but not limited to:

- Planning topology and/or layout of models
- Director's shot review tool
- "Whiteboard" and assignment review tool for educators

Preparing to Draw

1. The first step when using Grease Pencil is to enable the display of Grease Pencil drawings for the relevant view. To do this, locate the 'Grease Pencil...' entry in the 'View' menu; click on the 'Use Grease Pencil' toggle that appears in the panel.
2. At this point, click on 'Add New Layer' to add a new layer to draw on. This step is not necessary when you are starting off a new drawing (as a new layer will automatically be created for you), unless you want to customise the line width, colour, and opacity before drawing. However, if you want to draw on a new layer once layers already exist, it is necessary to click on the button.

Grease Pencil is available for the following spacetypes− 3d−View, Nodes Editor, Image Editor, Sequence Editor.

***Please note:* Grease Pencil data is currently unique to the particular screen area that it was created for, which means that if you remove that view, you lose your grease pencil data.**

Drawing

Quick Usage (For Just A Few Strokes)

- **To draw a stroke:** While holding Shift−LMB, start dragging the mouse to draw a new stroke. The stroke will finish when you release the mouse button.
- **To erase stroke(s):** While holding Alt−RMB, start dragging the mouse to erase segments of strokes that fall within the radius of the eraser 'brush'.

Easier usage (for drawing more complex sketches)

1. Enable the 'Draw Mode' toggle in top right−hand corner of the 'Grease Pencil' panel.
2. As for quickly drawing a few strokes, use the same mouse buttons to draw and erase, BUT without needing to use the modifier keys too (i.e. LMB to draw, RMB to erase).

Special Tricks in 'Draw Mode'

- **Drawing a straight line:** Hold CtrlKey while dragging with the LMB to draw a straight line. Although a wavy line will still appear on screen, only the endpoints of that stroke will be used for the final stroke that gets stored. This is a useful feature for architectural uses.
- **Drawing a dot:** Simply click on a spot. This is mentioned here because it is not available when 'Draw Mode' is not enabled.

For Tablet Users

- The thickness of a stroke at a particular point is affected by the pressure used when drawing that part of the stroke
- The 'eraser' end of the stylus can be used to erase strokes too

Sensitivity When Drawing

The default settings for the sensitivity to mouse/stylus movement when drawing, have been set so that there shouldn't be too much jitter while still allowing for fine details to be made. However, sometimes these settings may not be appropriate, in which case, the defaults can be found in the User Preferences under Edit Methods.

- **Manhatten Distance:** The minimum number of pixels the mouse should have moved either horizontally or vertically before the movement is recorded. Decreasing this should work better for curvy lines
- **Euclidean Distance:** The minimum distance that mouse should have travelled before movement is recorded. Imagine this as length of imaginary string between last recorded position and mouse−cursor.
- **Eraser Radius:** This is self−explanatory. It is simply the size of the eraser 'brush', so changing this will affect how likely strokes are going to be covered within the eraser brush and thus erased
- **Smooth Stroke:** This turns the post−processing step of smoothing the stroke to remove jitter. It is only relevant when not drawing straight lines. By default this is off, as it can often cause 'shrinking' of drawings, which is sometimes not that desirable.

Additional notes

- When 'Draw Mode' is enabled, many of the other events that are attached to the LMB and RMB are blocked.
- If 'Swap mouse buttons' is enabled, this has no effect on the mapping of mouse−buttons to drawing/erasing operations. However, it may become difficult to select using Shift−LMB in such a situation, in which case the tiny 'Lock' icon beside the 'Draw Mode' button should be enabled to help alleviate the problems (that will simply disable drawing from occurring with Shift−LMB).

Drawing Planes

Sketches are only relevant for the view/view−angle (referred to here as the 'drawing plane') that they were drawn at. There are several different options for how individual strokes (determined by the settings in use when the stroke was created) will be displayed.

- **Screen−Aligned:** This is the default drawing plane for the 3D−View, and is also the viewing plane that gets used for the other editors when 'Stick to View' is disabled. All new strokes that get drawn in this mode appear to be 'stuck' to the screen−surface (much like markers on a monitor), and will therefore stay unaffected by zooming/translating/rotating the view
- **View aligned (default for all 2D Views):** New strokes are affected by manipulating the view. This can be turned on/off using 'Stick to View' option.
- **Drawing in 3D−Space (only available in the 3D−View):** New strokes are drawn in 3D−space, with the position of their points being determined by the position of the 3D−cursor and the view rotation at the time.

Layers

Grease Pencil sketches are organised in layers, much like those you could find in the GIMP or Photoshop. These layers are not related to any of the other layer systems in Blender, and also do not have an upper limit on the maximum number of layers that can be used. Like the layers in the aforementioned apps, these layers can also be renamed, locked, hidden, and deleted.

Their main purpose is to collect together a bunch of sketches that belong together in some meaningful way (i.e. "blocking notes", "director's comments on blocking", or "guidelines"). For this reason, all the strokes on a layer (not just those made after a particular change) are affected by that layer's colour, opacity, and stroke thickness settings.

By default, most operations occur only on the 'active' layer. The active layer can be identified as the one with the different panel colour (in the default set, an light orangy−brown colour). Clicking on a layer, or changing any of its settings will make it the new active layer.

The active layer can also be identified by looking at the status indicator (in the top right−hand corner of every view with Grease Pencil data being shown). Animated Sketches

Grease Pencil can be used to do basic pencil tests (i.e. 2D animation in flipbook style). Sketches are stored on the frame that they were drawn on, as a separate drawing (only on the layer that they exist on). Each drawing is visible until the next drawing for that layer is encountered. The only exception to this is the first drawing for a layer, which will also be visible before the frame it was drawn on.

Therefore, it is simple to make a pencil−test/series of animated sketches:

1. Go to first relevant frame. Draw.
2. Jump to next relevant frame. Draw some more.
3. Keep repeating process, and drawing until satisfied. Voila! Animated sketches.

Onion Skinning

Onion−skinning (also known as ghosting), is a useful tool for animators, as neighboring frame(s) are lightly drawn by Blender. It allows animators to make judgments about movements, by comparing movement from different frames.

Usage Notes:

- Onion−skinning is enabled per layer by clicking on the 'Onion Skinning' button.
- The 'GStep' field controls how many frames will be drawn. When 'GStep' is 0, only the drawing on either side of the current frame will be visible. Otherwise, it this field specifies the maximum number of frames on either side of the current frame that will result in a neighboring drawing being included.

Adjusting Timing of Sketches

It is possible to set a Grease−Pencil block to be loaded up in the Action Editor for editing of the timings of the drawings. This is especially useful for animators blocking out shots, where the ability to re−time blocking poses is one of the main purposes of the whole exercise.

1. In an Action Editor window, change the mode selector (found beside the menus) to 'Grease Pencil' (by default, it should be set to 'Action Editor').
2. At this point, the Action Editor should now display a few 'channels' with some 'keyframes' on them. These 'channels' are the layers, and the 'keyframes' are the frames that each layer has. They can be manipulated like any other data in the Action Editor can be.

All the available Grease−Pencil blocks for the current screen layout will be shown. The Area/Grease−Pencil datablocks are drawn as green channels, and are named with relevant info from the views. They are also labelled with the Area index (which is currently not shown anywhere else though).

Copying Sketches

It is possible to copy sketches from a layer/layers to other layers in the Action Editor using the Copy/Paste buttons in the header. This works in a similar way as the copy/paste tools for keyframes in the Action Editor.

Sketches can also be copied from one screen (or view) to another using these tools. It is important to keep in mind that keyframes will only be pasted into selected layers, so layers will need to be created for the destination areas too.

Converting Sketches to Other Forms

In the 3D−view, sketches on the active layer can be converted to geometry, based on the current view settings. Sketches are converted into geometry by transforming the points recorded when drawing (which make up the strokes) into 3D−space (based on the current view settings). Currently, all points will be used, so it may be necessary to simplify or subdivide parts of the created geometry for standard use.

Sketches can currently be converted into one of three types:

- **Armature:** Each stroke is converted into a bone chain, which is assigned to an armature named after the active layer. The bones in each chain are connected and parented to each other. Also, bones inherit their envelope radii from the thickness of their stroke at each recorded point.
- **Bezier Curve and Path:** Each stroke is converted into a separate curve within a curve object that's named after the active layer. Handles are automatically set to be 'free' handles (i.e. the black type), and are set to be in the same places as the control−points. The weight/radius of the curve at each control−point is set to equal the thickness of the stroke at each recorded point. However, in order to see that, you need to set the 'BevOb' field to use a CurveCircle or similar curve.

Previous: Manual/LocalGlobal	Contents	*Next:* Manual/Manipulation in 3D Space

User Manual: Contents | Guidelines | Blender Version **2.4x**

Description

There are many things you can do with the selected object within your virtual world. You can twist it, make it bigger or smaller, spin it around, change its shape, and so on. This section tells you how to do these things to your objects.

As you will find with most of Blender, there isn't just "one way" to do things. For infrequent users, there is always the context−sensitive menus. For more experienced users, there are hotkeys, where merely tapping a key performs an action.

Menu Options

With an object selected in a 3D view, the menu bar shows the selections View, Select and Object. Click the Object selection to manipulate the object. The menu that pops up has the option Transform. Hovering over Transform pops up a sub−menu, showing you selections (and hot keys on the right) for manipulating the object:

- Grab/Move
- Rotate
- Scale
- Convert to Sphere
- Shear
- Warp
- Push/Pull

Since there are also hotkeys to do these things, please read on to the next section.

Previous: Manual/Layers	Contents	*Next:* Manual/Hotkeys

User Manual: Contents | Guidelines | Blender Version **2.4x**

Grab/Move

Mode: Object Mode/Edit Mode

Hotkey: G

Menu: Object/Mesh/etc Transform Grab/Move

Description

One of the fastest ways to move things in 3D space is with G. Pressing the hotkey will enter the 'grab/move' transformation mode where the selected object or data is moved according to the mouse pointer's location. The distance from the mouse pointer to the manipulated object has no effect.

Options

LMB
: Confirm the move, and leave the object or data at its current location on the screen

MMB
: Constrain the move to the X, Y or Z axis.

RMB or Esc
: Cancel the move, and return the object or data to its original location

Proportional Edit
: The circle of influence, changed by MW U to make smaller and MW D to make bigger, is shown and will move vertices within that circle by the proportional method/falloff chosen.

See the rest of the Manipulation in 3D space section for additional options available within the transformation modes.

Rotate

Mode: Object Mode/Edit Mode

Hotkey: R

Menu: Object/Mesh/etc Transform Rotate

Description

Pressing R will enter the 'rotate' transformation mode where the selected object or data is rotated according to the mouse pointer's location. This mode uses the angle from the pivot point to the mouse pointer as the angle for rotation, so moving the mouse further from the object/data will give more fine−grained precision (i.e. the mouse movement will affect the rotation less, for the same mouse distance moved).

Options

LMB
: Confirm the rotation, and leave the object or data at its current rotation on the screen

MMB
: Constrain the rotation about the X, Y or Z axis.

RMB or Esc
: Cancel the rotation, and return the object or data to its original rotation

R (Trackball)
: Pressing R while already rotating toggles the rotation mode between a single axis rotation (either aligned to the screen or around a certain axis) and a two axis, 'trackball' rotation. In Trackball mode, the rotation of the object is controlled by both the X and Y location of the mouse pointer, similar to the trackball view rotation mode. This can be a quick way to rotate an object into place, without having to keep changing the view rotation while adjusting.

See the rest of the Manipulation in 3D space section for additional options available within the transformation modes.

Scale

Mode: Object Mode/Edit Mode

Hotkey: S

Menu: Object/Mesh/etc Transform Scale

Description

Pressing S will enter the 'scale' transformation mode where the selected object or data is scaled inward or outward according to the mouse pointer's location. The object/data's scale will increase as the mouse pointer is moved away from the pivot point, and decrease as the pointer is moved towards it. If the mouse pointer crosses from the original side of the pivot point to the opposite side, the scale will continue in the negative direction, making the object/data appear flipped. The precision of the scaling is determined by the distance from the mouse pointer to the object/data when the scaling begins

Options

LMB
: Confirm the scale, and leave the object or data at its current scale on the screen

MMB
: Constrain the scaling to the X, Y or Z axis.

RMB or Esc
: Cancel the scale, and return the object or data to its original scale

Alt S
: Scales along the selected normal direction

See the rest of the Manipulation in 3D space section for additional options available within the transformation modes.

Precision

Mode: Object Mode/Edit Mode

Hotkey: Ctrl, Shift

Description

Holding Ctrl and/or Shift during a manipulation can be used to control the precision more finely.

Options

Ctrl (Snap)
: Snap the transformation on 1 blender unit (or 1/10 blender unit depending of the zooming view)

Shift (Precise)
: Allow more finer transformation control but not by precise values

Ctrl and Shift
: Snap the transformation on 1/10 blender unit (or 1/100 blender unit depending on the zooming view)

Numeric transformations

Mode: Object Mode/Edit Mode

Hotkey: G, R, S then NumPad +/− and NumPad 0−9 or Keyboard 0−9

Description

When doing a tranformation, instead of using the mouse (imprecise work), you can directly pass a precise value.

Hit NumPad − (on a french keyboard the minus is under the number 6 key so you must use the Numpad) if you want negative values then a numeric value. You can see the values in the header of the 3D View. If you were using an axis constraint (global or local), the value is applied to that axis. If there is no axis constraint and you want to choose which axis to control, you can use Tab to switch from any of the axis (a cursor appears after the value). You can also return to a mouse control with Backspace

Cancel transformations

Mode: Object Mode

Hotkey: Alt G, Alt S, Alt R

Menu: Object Clear/Apply Clear Location, Clear Scale, Clear Rotation

Description

You can clear any transformation done in Object Mode.

Options

AltG
: Clear the location of the selection

AltS
: Clear the scale of the selection

AltR
: Clear the rotation of the selection

Align

Mode: Object Mode

Hotkey: Ctrl Alt A

Menu: Object Transform Align to Transform Orientation

Description

Align selected objects to a specific Transform Orientation.

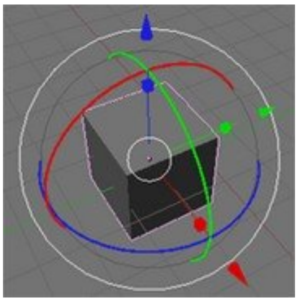
Previous: Manual/Manipulation in 3D Space	Contents	*Next:* Manual/Manipulators

User Manual: [Contents] | [Guidelines] | Blender Version **2.4x**

Manipulators

Mode: Object mode, Edit Mode

Hotkey: Ctrl Space



Combination manipulator

When using the normal Transform commands (G=Grab, R=Rotation, S=Scale) they will work only parallel to the current view. (The transform commands can be augmented with additional modifer keys. E.g. pressing X, Y or Z immediately after G, R, or S will contrain the transform to that global axis (using MMB will do the same but along the nearest axis from the mouse pointer). Pressing R, a second time before pressing LMB (i.e. while the rotation command is in progress, releases the default "parallel" rotation constraint, allowing free rotation in any axis. (This is only noticeable in a non "user" view (Front, Right, Top ..etc)

Read more about Axis Locking

Manipulators provide a visual representation of the transform commands and allow movement, rotation and scaling along any axis in any mode of the 3d window.

Manipulator Header

The manipulator can also be accessed in the header of the 3D View window (Shift LMB can select more than one manipulator at the same time)

Hand
: Enable/disable the manipulators

Triangle
: Translation/location

Circle
: Rotation

Box
: Scale

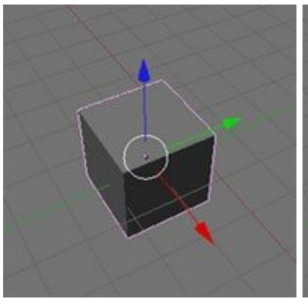
Tranform Orientation
: Control the orientation of the axis for transformations

Manipulator types

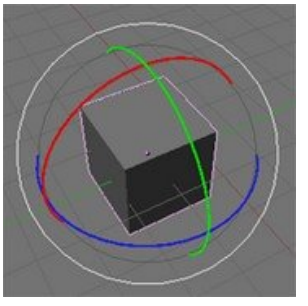
Mode: Object mode, Edit Mode

Hotkey:
Ctrl Alt G Translate / Move
Ctrl Alt R Rotate
Ctrl Alt S Scale

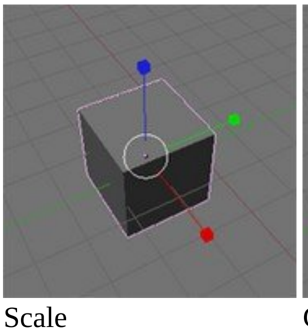
There is a separate manipulator for each Transform Command. Each can be viewed separately or in a combination.



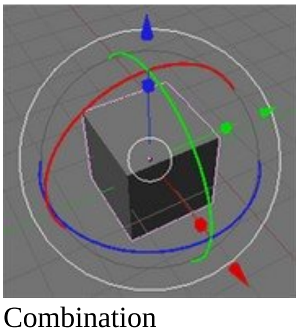
Translate



Rotate



Scale



Combination

Other Manipulator controls

Holding down Ctrl constrains the action in certain increments (Loc/scale = 1 Grid unit, Rot= 5 degrees)

Holding down Shift when you LMB on one of the handles the Manipulator action will be performed on the other two axes to the one you Clicked on, you can let go of Shift once you have LMB

LMB on the white circle (largest circle around rotation Manipulator) do the same as pressing R

LMB on the grey circle (inner circle around rotation Manipulator) do the same as pressing R twice (tracking rotation)

Manipulator Preferences

Manipulator preferences

The settings of the Manipulator eg. size can be found in the 'View & Controls' section of the user preferences window

Size
: Diameter of Widget, in 10 pixel units

Handle
: Size of widget Handles, as a percentage of widget radius (size/2)

Hot spot
: Hotspot size for clicking Widget Handles

Transform Orientation

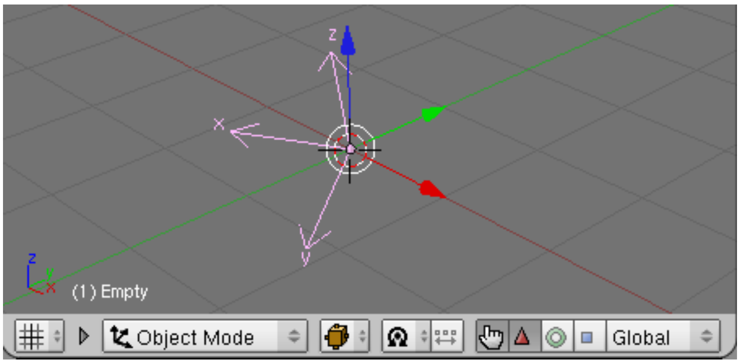
Mode: Object Mode, Edit Mode

Hotkey: Alt Space

The default (global) transform manipulator is very useful but in some situations it's not. e.g.: scale a rotated object along the rotated axis. Luckily Blender can change the orientation of the Transform Manipulator.

Below is a list of different manipulator types. On every image, compare the position of the manipulator axes (color axes over the object) with the global (lower left corner of the 3D window) and local (the object is a empty, so just the local axes of the object are shown) ones.

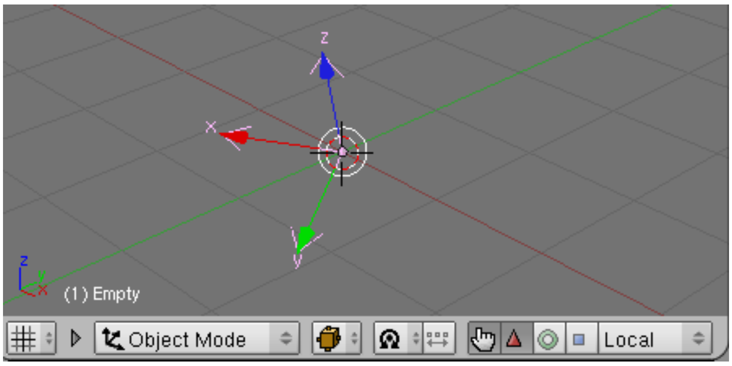
Global



Global

Global − The manipulator matches the global axis.

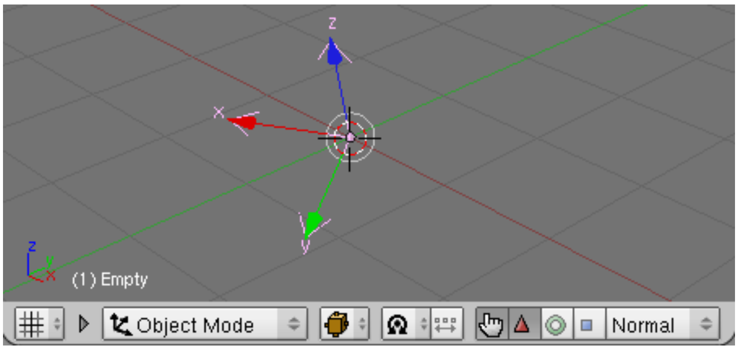
Local



Local

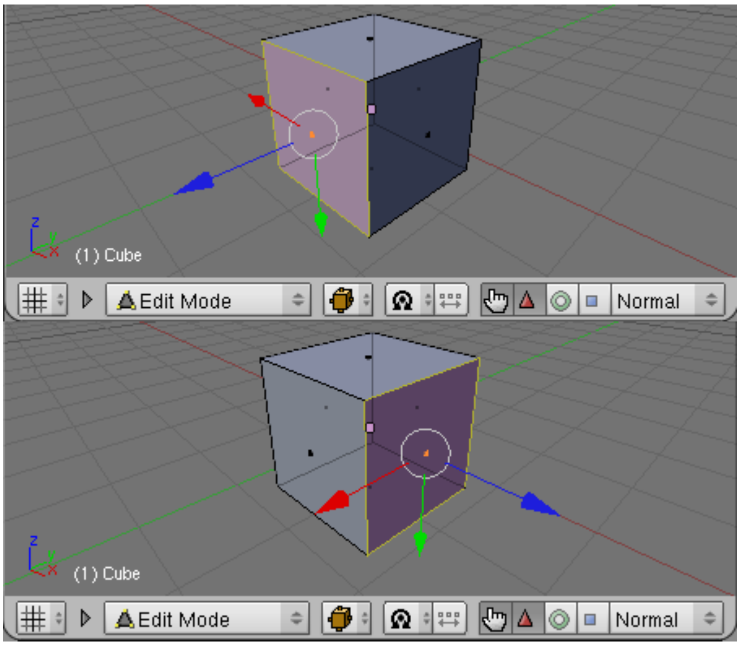
Local − The manipulator matches the object axis.

Normal



Normal

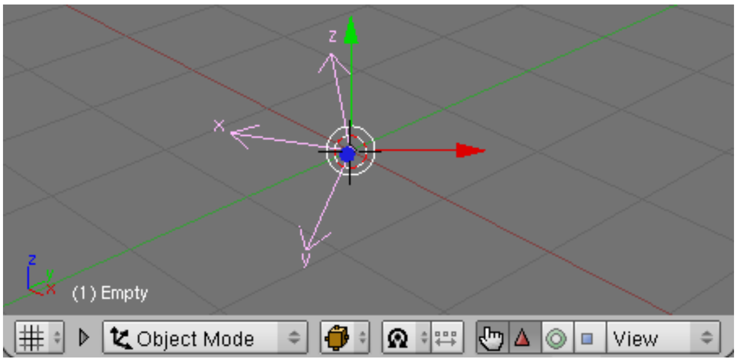
Normal − The Z axis of the manipulator will match the normal vector the selected object. Not very useful for the empty. See the example below.



Cube example

A better example using a object with face normals, selecting faces on edit mode:

View



View

View − The manipulator will match the 3D window, Yâ†’UP/DOWN, Xâ†’Left/Right, Zâ†’towards/away from you.

Previous: Manual/Hotkeys	Contents	*Next:* Manual/Gestures

User Manual: Contents | Guidelines | Blender Version **2.4x**

Gestures

Mode: Object Mode / Edit Mode

Hotkey: LMB (drag)

Blender's 3D transform modes can also be invoked by drawing mouse gestures. The tool is designed to figure out what mode to enter based on a hand−drawn gesture. After a gesture is drawn as described below, release the LMB . Move the mouse without pressing any button, then click the LMB when you achieve the effect you want. To cancel, click the RMB , even if movement in the scene has occurred.

There are three gestures the tool recognizes:

- Scale
- Translate
- Rotate

Scale

Example Scale gesture.

To activate Scale mode draw what appears to be a **V** shaped path using the LMB . (*Example Scale gesture*) is an example of a hand−drawn **V**. It doesn't have to be exact but the closer and sharper it is the better the tool will understand. If the **V** has some roundness in it it most likely will be interpreted as a request for Rotate mode.

Rotate

Example Rotate gesture.

To activate Rotate mode draw what appears to be a **C** shaped curve using the LMB . (*Example Rotate gesture*) is an example of a hand−drawn **C**. It doesn't have to be exact but the smoother the curve the better the tool will understand. If the **C** has a sharp corner in it it most likely will be interpreted as a request for Scale mode.

Translate

Example Translate gesture.

To activate Translate mode draw what appears to be a − or line using the LMB . (*Example Translate gesture*) is an example of a hand−drawn −. It doesn't have to be exact but the straighter the line the better the tool will understand. If the − deviates too much from a straight line it most likely will be interpreted as a request for either Scale mode or Rotate mode.

Previous: Manual/Manipulators	Contents	*Next:* Manual/Axis Locking

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.4x**

Normal axis locking

Mode: ObjectMode / EditMode (translate, rotate, scale, extrude)

Hotkey: X, Y, Z

Blender has a very useful option: If you want to

- S − Scale
- R − Rotate
- G − Move/Grab/Translate
- E − Extrude

the selected object, the operation can be constrained to one axis:

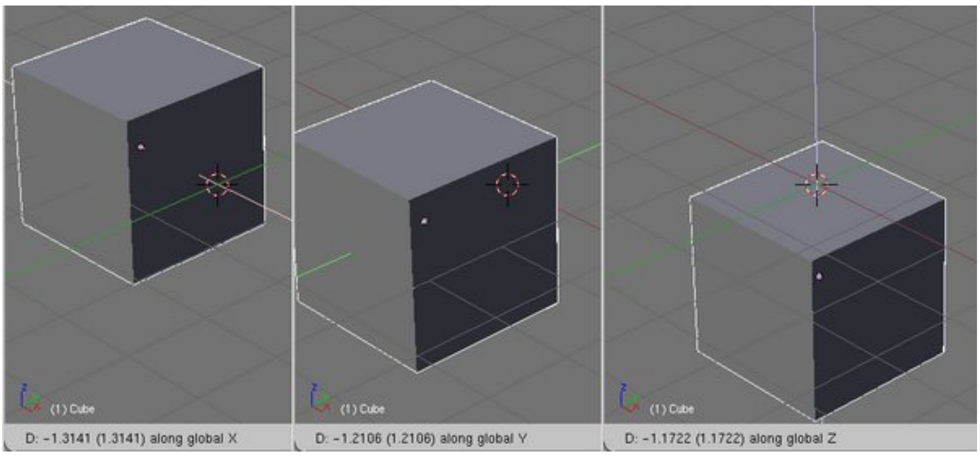
Press the appropriate key to start the operation (i.e. S to scale), then press X, Y, or Z to constrain to the corresponding global axis.

If the same "axis" key is pressed again, the operation is constrained to the object's Local axis e.g. G, X, X to constrain to the object's local X axis.

Move/Translate

Press :

- G,X for the global X−axis (left and right).
- G,Y for the global Y−axis (forward and back).
- G,Z for the global Z−axis (up and down).

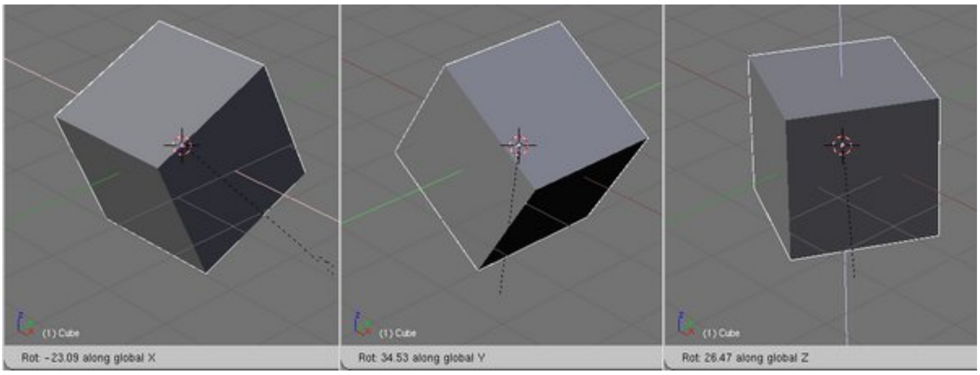


Translation/Grab with locked axis X, Y and Z

Rotate

Press:

- R,X for the global X−axis (from left to right).
- R,Y for the global Y−axis (from front to back).
- R,Z for the global Z−axis (up and down).

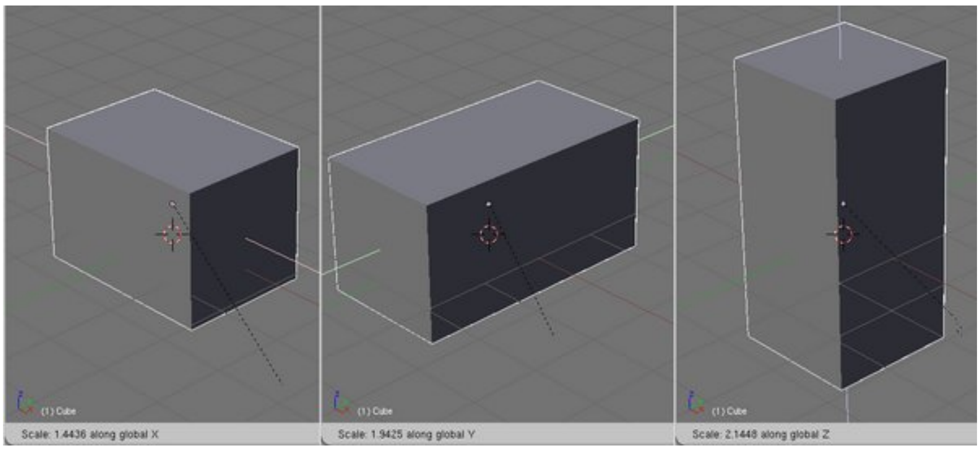


Rotation with locked axis X, Y and Z

Scale

Press :

- S,X for the global X−axis (left and right).
- S,Y for the global Y−axis (forward and back).
- S,Zfor the global Z−axis (up and down).



Scale with locked axis X, Y and Z

Plane axis locking

Mode: ObjectMode / EditMode (translate, rotate, scale, extrude)

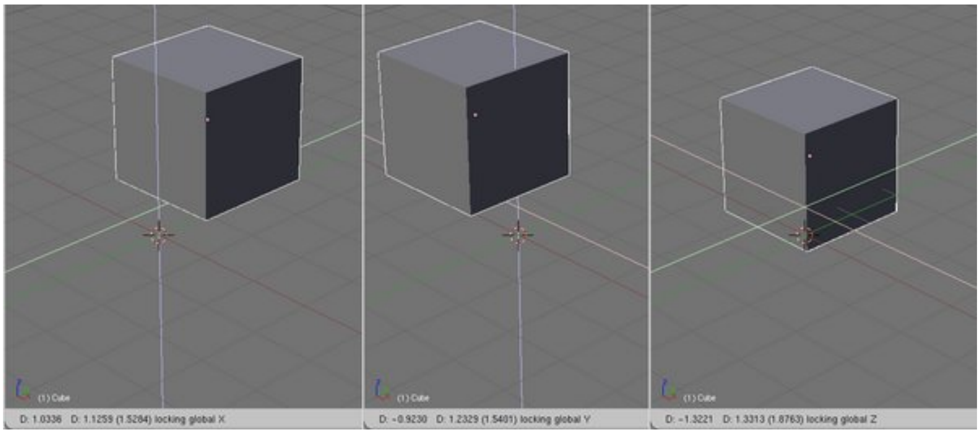
Hotkey: Shift X, Shift Y, Shift Z

You can lock two axes at once at both the global and the local axes.Press Shift X, Shift Y or Shift Z to lock the OTHER two axes. i.e

- Shift X − Uses Y+Z
- Shift Y − Uses X+Z
- Shift Z − Uses X+Y

Move/Translate

- Press G,Shift X to move only along the global Y−axis and Z−axis (not left and right)
- Press G,Shift Y to move only along the global X−axis and Z−axis (not forward and back).
- Press G,Shift Z to move only along the global X−axis and Y−axis (not up and down).



Translation/Grab with locked axes Y+Z, X+Z and X+Y

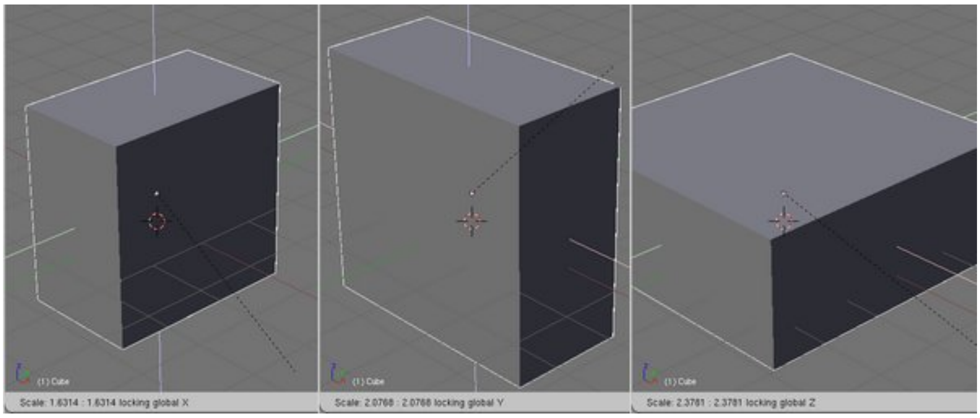
Rotate

For rotation pressing Shift has no effect (except for the display of different axes).

Rotating an object by locking two axes has the same effect as using only one axis to lock it, as rotation using two axes will rotate the object on the unlocked axis.

Scale

- Press S,Shift X for the global Y−axis and Z−axis (not left and right)
- Press S,Shift Y for the global X−axis and Z−axis (not forward and back).
- Press S,Shift Z for the global X−axis and Y−axis (not up and down).



Scale with locked axes Y+Z, X+Z and X+Y

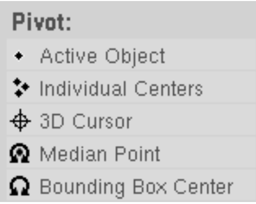
Previous: Manual/Gestures	Contents	*Next:* Manual/Pivot Points

User Manual: Contents | Guidelines | Blender Version **2.45**

Pivot

Mode: Object Mode and Edit Mode

Menu: Droplist in the menu of the 3D view



Pivot Point Modes

The pivot point is the point in space around which all rotations, all scalings and all mirror transformations are centered. We can chose among five general modes for our pivot points which can be selected from a drop list on the header of any 3D area as seen here in *Pivot Point Modes*. Our job is to chose the most efficient type for the task and to position pivot point accurately.

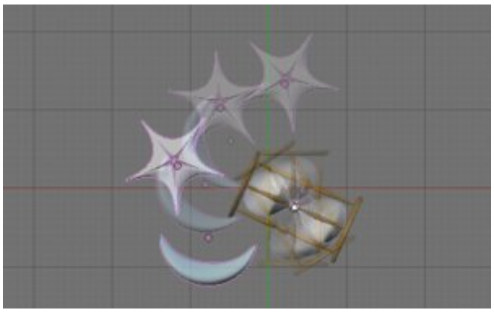
Active Object

Mode: Object Mode, Edit Mode

Hotkey: Alt .

Forget the name: it is not limited to Objects. In Edit Mode the *active* element can also be a vertex, an edge or a face.

In Object Mode



Rotation around the active Object.

What happens in Object Mode is pretty simple: rotations and scalings happen around the active Object's Center. This is illustrated by *Rotation around the active Object* where the Center of the active Object, and hourglass, is the only thing not to remain still.

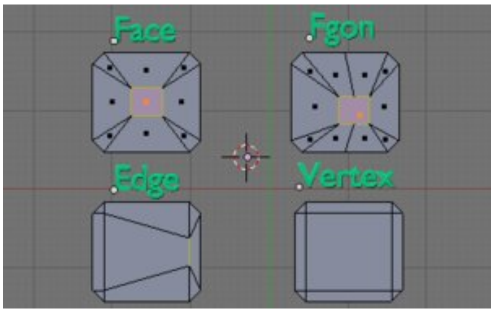
In Edit Mode

It may seem like there is a lot of complexity to using the active element as a pivot point in Edit Mode but, in reality, the many possibilities result of only a few rules:

- the pivot point is always at the median of the active element(s);
- the transformations occur by transformation of the **vertices** of the selected element(s). If an unselected element shares one or more vertices with a selected element then the unselected one will get some degree of transformation also.

Let's examine the following examples: in each case we will see that the two rules apply.

Single selection

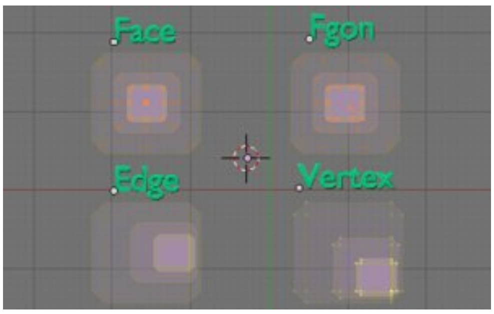


Only one element selected.

When one single element is selected it becomes automatically active. In *Only one element selected* we can see that when it is transformed its vertices move with with the consequence that any adjacent element which shares one or more vertices with the active element is also transformed somewhat.
Lets review each cases:

- *Faces* have their pivot point where their selection dot appears which is where the median of its vertices.
- *Fgons* behave the same but notice that the selection dotcan be off.
- *Edges* have their pivot point on their middle since this is always where the median of an edge is.
- A single *Vertex* has no dimensions at all so it can't show any transformation.

Multiple selection



Edit Mode and multiple selections.

When multiple elements are selected they all transform.

The pivot points stay in the same place as what we've seen above, with only one exception for Fgons. In *Multiple selections*the selected elements have been scaled down from large to small:

- For *Faces* the transformation occurs around the selection dot of the active face.
- *Fgons* behave like faces with one exception that can be seen on *Multiple selections*: when the Fgon is completely surrounded by selected faces it simply cannot be made active (writing this on Jan 5 2008). The center pivot is then the median of all selected elements.
- *Edges* also keep the same behavior with their pivot point at their median.
- There is a case for *Vertices* this time: the active Vertex is where the pivot point resides. All other vertices are transformed relative to it.

Again, as we have seen, all there is to remember is the two rules:

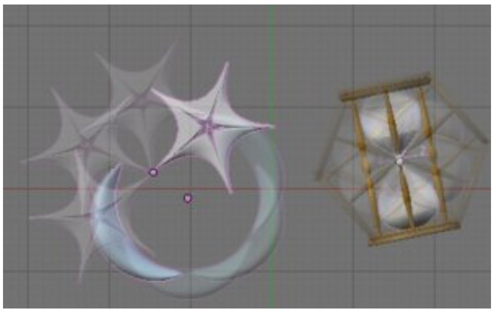
- pivot point at the median of the active element's vertex or vertices;
- all selected vertices, directly or as part of a bigger element, e.g. a face, and only them are transformed.

Individual Objects Center

Mode: Object Mode, Edit Mode

Hotkey: Ctrl .

In Object Mode



Rotation around the individual centers.

In *Rotation around the individual centers.* the Object centers of each Object remains at the same location while each Object is rotating around them.

Positioning the center of the Objects is a most useful technique that affords us more control over our animations. Let's examine *Rotation around the individual centers.*:

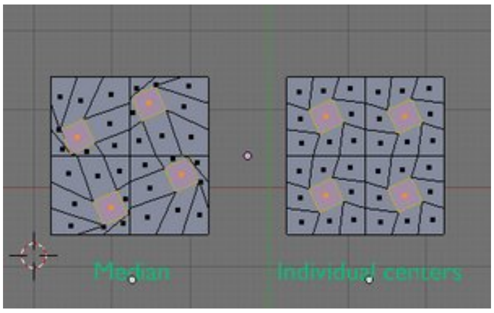
- the center of the hourglass is coincident with the median of all its components;

- the positioning of the center of the star at the tip of one of its branch allow for a rotation around it with only the use or Rot Ipos that are more faithfully interpolated than what we get using the 3D cursor in the same position.
- the center of the crescent is completely outside of what to us appears to be the Object. The user must understand that the center really marks **where the Object is**; what we see on the screen is a description of what the object is made of: vertices, colors, stuff and it can very well happen happen to be off−center, like for the crescent here.

In Edit Mode

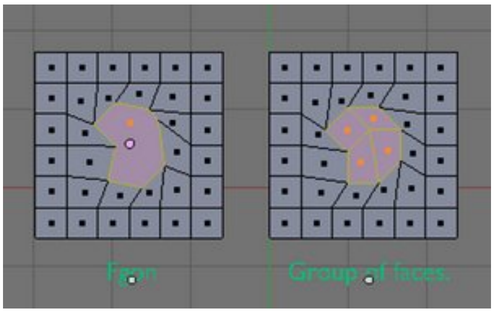
With the Vertex or the Edge selection methods in use, a selection of vertices or one of edges has its pivot point at the median of the set of vertices so selected. For more information see the Median Point pivot section.

As soon as the Face selection method is in use though the pivot point as the center of those Faces becomes possible.



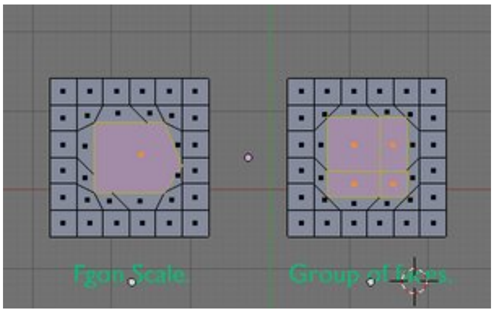
Individual rotation of multiple faces.

It is possible to rotate individually each face only in Face selection mode. Only faces that don't touch can be transformed in this way without deforming. We cannot use the Proportional Editing Tool (PET) while transforming individual faces this way.



Fgon rotation with individual centers pivot point.

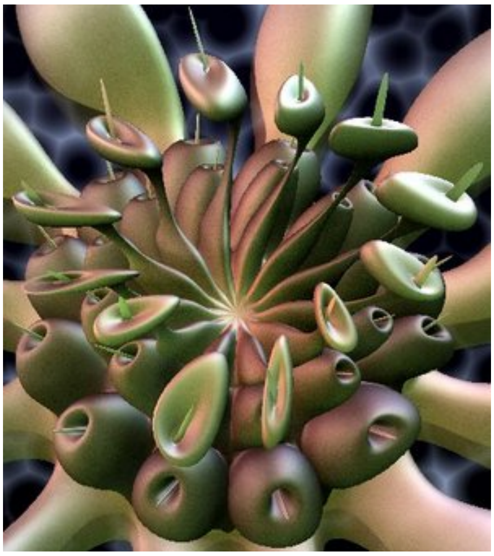
Faces that touch, even when they are inside an Fgon, are deformed when rotated with individual centers as the pivot point.



Problems with Fgon and groups of faces scaling.

Fgons and group of Faces can be scaled and their outside perimeter won't be deformed. The individual faces inside them aren't uniformly scaled though, something one must take into account.

All those deformations won't happen if one is not using the Face selection mode; it becomes impossible to edit more than one face or one group of faces at a time though.



Modelisation with faces and individual centers pivot point.

Once one is aware of its limitations and pitfalls this peculiar tool can save a lot of time and lead to unique shapes. This 'anemone' was modeled from a 12 sided cylinder in about 10 minutes by using repeatedly this workflow: extrusions of individual faces, scaling with *median as a pivot point*, and scaling and rotations of those faces with *individual centers as pivot points*.

3D Cursor

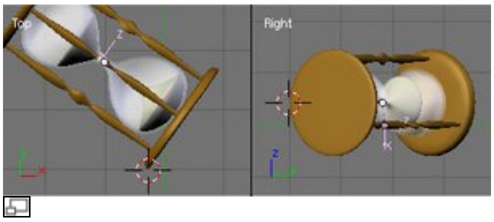
Mode: Object Mode, Edit Mode

Hotkey: .

The 3d cursor is the simplest most intuitive of all pivot points; it allows for total control of the results. It can be summarize by this: position it and transform.

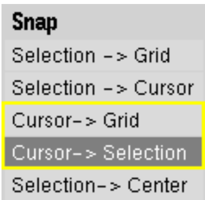
Positioning the 3D cursor

There are a few methods to position the 3D cursor.

- 

  Positioning the 3D cursor with two orthogonal views.

  Using LMB in the 3D area. For accuracy one must look and use two orthogonal (perpendicular) 3D areas, Any combination of top (NumPad 7), front (NumPad 1) and right side (NumPad 3) is the easiest to access. That way, in one view one can control the positioning along two axes and determine depth in the second view.

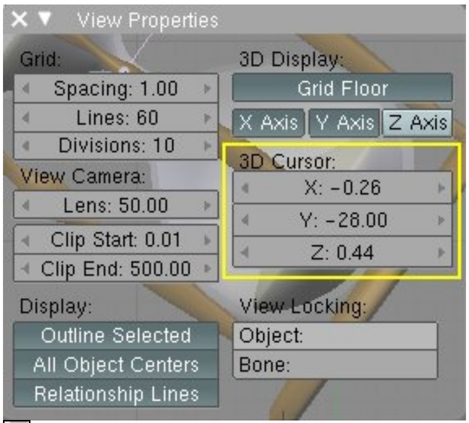
- 

  The snaps dialog.

  Using snaps:

  - Shift S Cursor −> Grid to send the 3D cursor to the nearest **visible** point of the grid;
  - Shift S Cursor −> Selection to send the 3D cursor to :
    - the Object Center of an object;
    - to a vertex
  - when there is more than one element in the selection and the bounding box pivot point is selected, Shift S sends the 3D cursor to:
    - in Object Mode, to the **bounding box** surrounding the Objects centers;
    - in Edit mode, to the center of the bounding box surrounding the selected vertice (even in Edge selection mode or in Face selection mode it really is the vertice that are selected indirectly that are taken into account.
  - when the **median** pivot point is selected, Shift S sends the 3D cursor to:
    - in Object mode, the median surrounding the Object centers ;
    - in Edit mode, to the the median of the selected vertice

Lots of possibilities = lots of power.

- 

  The View Properties dialog

  Numerically, using the View Properties menu entry (View > View Properties) from the 3D View header bar, and then entering the 3D Cursor location in the 3D Cursor section of the resultant dialog box that should now be visible.

Transformation

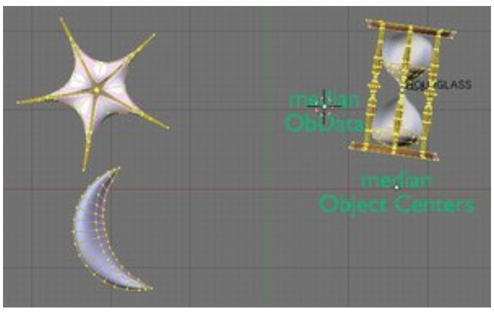
All there's left to do is to select the 3D Cursor as the pivot point and rotate, scale or mirror.

Median Point

Mode: Object Mode, Edit Mode

Hotkey: Ctrl ,

We can assimilate the Median point to the notion of Center Of Gravity (COG): supposing that every element of the selection has the same mass, the median point would sit at the COG, the point of equilibrium for the selection. This is very abusive as we will see soon enough. Yet it helps predicting where the Median point should be when planing a scene.



Median point of Object Centers and ObData.

In Object Mode

For Objects, only the Object Center is taken into account. Moreover, each Object Center is assumed to have the same mass. This can lead to very counter−intuitive results; On the *Median point of Object Centers and ObData* we see that the Median of the Objects sits at the middle of the Object centers. That is because the ObData (the geometry) of the moon and the star is way off their Object Center.

In Edit Mode

Still on *Median point of Object Centers and ObData* we see that even the position of the Median point for the ObData is surprisingly close to the hourglass: this is because it has much more vertice (611) than the moon (81) and the star (130). Blender supposes that every vertex has the same weight.

Transformation

Once the Median point has been chosen from the list the widget immediately cling to it, giving and excellent visual clue: all the rotations, scalings and mirror will happen around this point.

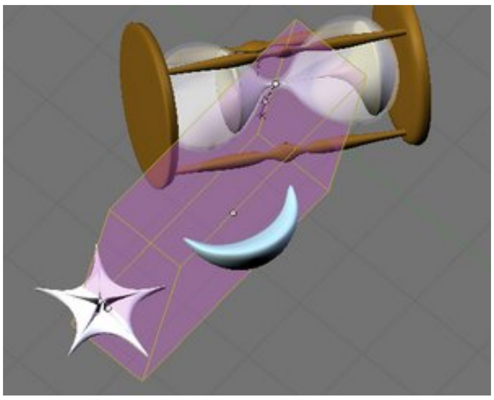
Bounding Box Center

Mode: Object Mode, Edit Mode

Hotkey: ,

The bounding box is a rectangular box that is wrapped as tightly as possible around the selection. It is oriented parallel to the World axes. In this mode the pivot point lies at the center of the bounding box.

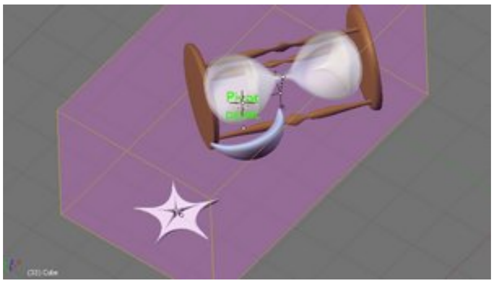
In Object Mode



The Bounding Box in Object Mode.

In Object mode the Bounding box is wrapped around the Object Centers and does not take into account the ObData (geometry)

In Edit Mode



The_Bounding_box_in_Edit_Mode.

This time it is the ObData that is enclosed in the box because all of its vertice were selected, The bounding box in Edit Mode takes no account of the Objects centers but only of the selected vertices.

Previous: Manual/Axis Locking | Contents | *Next:* Manual/Proportional Edit

User Manual: Contents | Guidelines | Blender Version **2.43**

Proportional Edit

Mode: Edit Mode

Hotkey: O / Alt O / Shift O

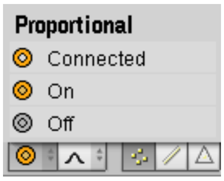
Menu: Mesh Proportional Editing

When working with dense geometry, it can become difficult to make subtle adjustments to the vertices without causing nasty lumps and creases in the model's surface. When you face situations like these, use the proportional editing tool. It acts like a magnet to smoothly deform the surface of the model, without creating lumps and creases, by also modifying unselected vertices within a given range, not just the selected vertices.

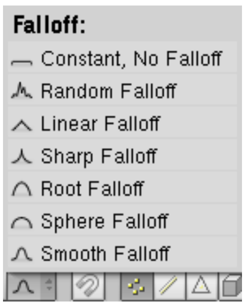
Sculpting

Blender offers a complete Sculpt Mode which contains a set of brushes and tools for proportionally editing a mesh without seeing the individual vertices.

Options



Proportional Editing icon



Falloff menu.

The Proportional Editing mode menu is available in Edit Mode on the 3D View header

Off
: O. Proportional Editing is Off, only selected vertices will be affected.

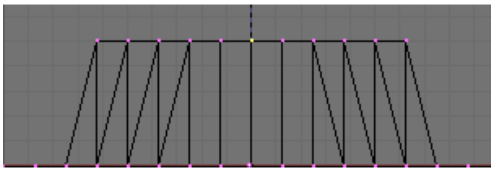
On
: O or Alt O. Vertices other than the selected vertex are affected, within a defined radius.

Connected

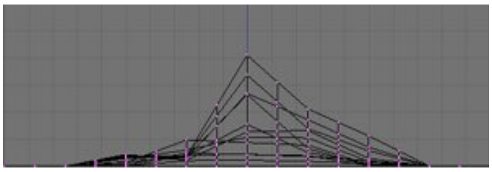
Alt O. Rather than using a radius only, the proportional falloff propagates through connected geometry. This means that you can easily proportionally edit the vertices in a finger of a hand, without affecting the other fingers, since although the other vertices are nearby spatially, they are far away following the topological edge connections of the mesh. The icon will be cleared, (grey), in the center when Connected is active.

Falloff

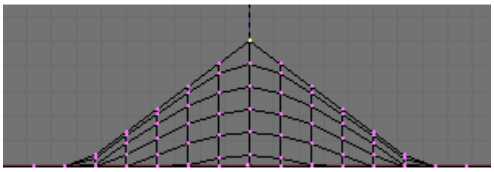
While you are editing, you can can change the curve profile used by either using the Mesh Proportional Falloff submenu, using the toolbar icon *Falloff Menu.* or by pressing Shift O to toggle between the various options.



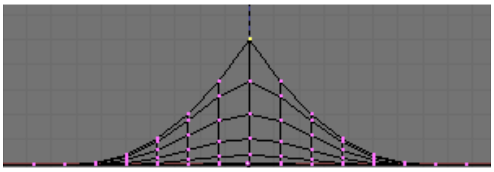
Constant − No Falloff.



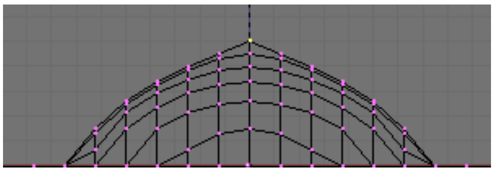
Random Falloff.



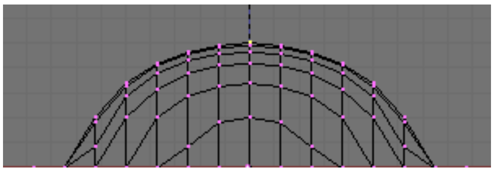
Linear Falloff.



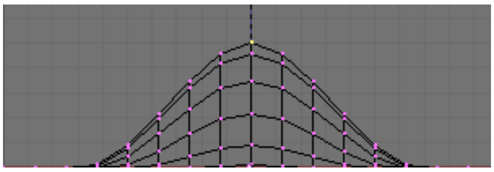
Sharp Falloff.



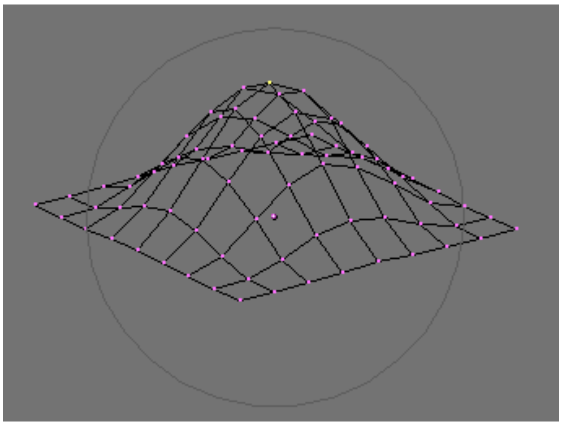
Root Falloff.



Sphere Falloff.



Smooth Falloff.



Influence circle.

Influence

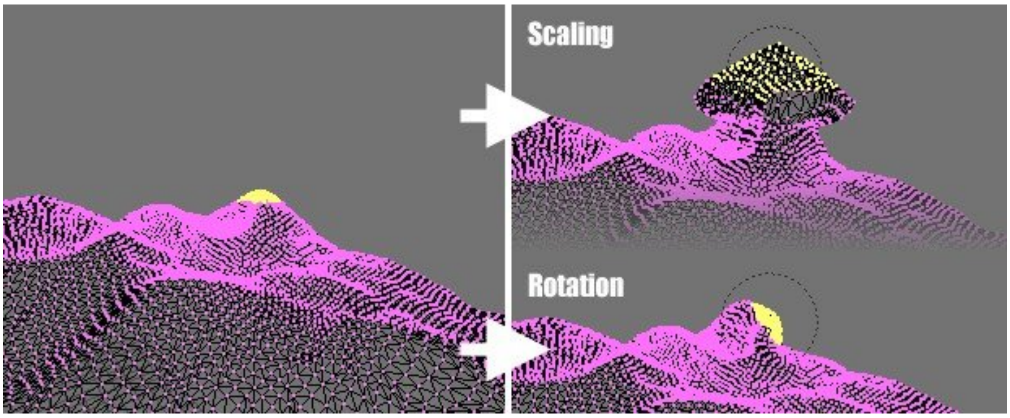
You can increase or decrease the radius of the proportional editing influence with the mouse wheel

MW or Page Up and Page Down respectively. As you change the radius, the points surrounding your selection will adjust their positions accordingly.

Examples

Switch to a front view (NumPad 1) and activate the grab tool with G. As you drag the point upwards, notice how nearby vertices are dragged along with it. When you are satisfied with the placement, press LMB to fix the position. If you are not satisfied, cancel the operation and revert your mesh to the way it looked before with RMB or Esc key.

You can use the proportional editing tool to produce great effects with the scaling (S) and rotation (R) tools, as *A landscape obtained via Proportional Editing* shows.



A landscape obtained via Proportional Editing

Combine these techniques with vertex painting to create fantastic landscapes. *Final rendered landscape* shows the results of proportional editing after the application of textures and lighting.

Final rendered landscape

Previous: Manual/Pivot Points | Contents | *Next:* Manual/Transform Orientations

User Manual: Contents | Guidelines | Blender Version **2.4x**

Transform Orientations

Mode: Object Mode, Edit Mode

Hotkey: Alt Space

A secondary orientation can be selected with Alt Space or through the Orientation menu in a 3D view header.

This orientation can then be used as a transformation constraint.

Axis Locking

Mode: Object Mode, Edit Mode

Hotkey: XX, YY, ZZ

Pressing the axis locking key twice locks to the user selected transform orientation.

Plane Locking

Mode: Object Mode, Edit Mode

Hotkey: Shift XShift X, Shift YShift Y, Shift ZShift Z

The same principle applies to locking movement on a plane. Press the plane locking keys twice to lock to your selected transform orientation.

Custom Orientations

Mode: Object Mode, Edit Mode

Hotkey: Shift Ctrl C

Custom Transform Orientations can be defined by the user using Object or Mesh elements. Custom Transform Orientations defined from objects use the local orientation of the object whereas those defined from selected mesh elements (vertice, edges, faces) use the normal orientation of the selection.

A name also needs to be assigned to the new orientation.

Note: When adding new orientations, if the name correspond to an already existing custom orientation, the new orientation will replace the old one.

[Demo Video] (XviD)

[Bits of Blender]'s has a video tutorial on this topic: [Youtube link]

Transform Orientations Panel

The Transform Orientations Panel, available from the *View* menu, can be used to manage Transform Orientations: Selecting the active orientation, Adding and Removing custom orientations and Clearing all custom orientations.

Previous: Manual/Proportional Edit	Contents	*Next:* Manual/Transform Properties

User Manual: Contents | Guidelines | Blender Version **2.4x**

Transform Properties Panel

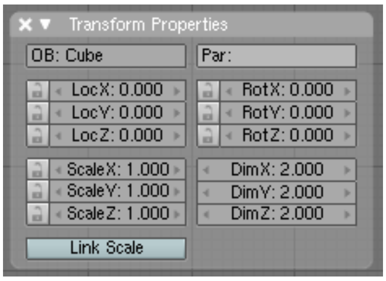
Mode: Edit or Object mode

Hotkey: N

Menu: Object ? Transform Properties

The Transform Properties popup panel is a modeless dialog meaning it can continue to be visible while you perform other activities. The dialog is actively updated. For example, as you rotate an object the *Rot* fields are updated in realtime.

Options



Transform Properties Panel (Object mode)

OB
: The object's name.

Par
: The name of the parenting object, if one is assigned. By entering in a name you are assigning a parent object. The name must match an existing object; if it doesn't then the name is erased from the field.

LocX, LocY, LocZ
: The object's center location in global coordinates relative to the object's center. see example.

RotX, RotY, RotZ
: The object's orientation relative to the object's center.

ScaleX, ScaleY, ScaleZ
: The object's scale relative to the object's center. Each object (cube, sphere, etc), when created, has a scale of one blender unit outward on each side of center. To make the object bigger or smaller, you scale it in the desired dimension.

DimX, DimY, DimZ
: The object's basic dimensions (in blender units) from one outside edge to another, as if measured with a ruler. For multi−faceted surfaces, these fields give the dimensions of a bounding box (think of a cardboard box) just big enough to hold the object.

Link Scale
: If this toggle−button is activated the relation of the X, Y and Z values in the Scale− and the Dim Fields is always preserved. Changing one value will change all the others as well with the same multiplication−factor.

Use this panel to either edit or display the object's transform properties such as position, rotation and/or scaling and this includes the object's name and parent assignment. These fields change the object's center and

then affects the aspect of all of its vertexes and faces.

IPO Note: The values of the location, rotation, and scale can also be affected by an IPO keyframe; so if there are IPO keys assocated with the object, be sure to reset them after making changes in this panel, or your changes will be lost when the frame is changed (IPO keys override manually−set properties).

Some fields have extra functionality or features, such as scroll regions. When attempting to edit these types of fields it is easier to use Shift LMB instead of just LMB . After you have edited a field click outside of the field's edit area or click ENTER to confirm the changes; changes will be reflected in the display window immediately. To cancel hit ESC.

Transform Properties Locking

The locking feature of the Location, Rotation and Size fields allow you to control a transform property solely from the properties panel. Once a lock has been activated any other methods used for transformation are blocked. For example, if you locked the LocX field then you can't use the mouse to translate the object along the object's X axis. However, you can still translate it using the LocX edit field. Consider the locking feature as a rigid constraint only changeable from the panel.

To lock a field click the padlock icon next to the field. The field is unlocked if the icon appears as (), and it is locked if the icon appears as ().

For further descriptions of the other features of an edit field see The Interface section.

User Manual: Contents | Guidelines | Blender Version **2.43**

Scene Management Structure

Scene management and library appending/linking is based on Blender's Library and Data System, so it is a good idea to read that manual page first if you're not familiar with the basics of that system.

Blender can be used to create something as simple as a single scene or image, or scaled up to an entire movie. A movie is usually comprised of three acts:

1. Introduction−Conflict
2. Rising Tension
3. Climax−Resolution

Each act contains a few scenes; settings where the action happens. Each scene is shot on a set, stage or location. Each is set with props and backdrops. The scene is a set of action sequences where the actors act (hopefully convincingly). Each sequence, or shot, usually lasts a few seconds.

Sequence shot

Sometimes, a single shot lasts several minutes: its a "sequence shot", which might even be a complete scene on its own. Technique hard to master if you don't want your audience to fall asleep!

A single Blender file is organized and set up to be able to contain an entire movie. Each blend file can contain multiple Scenes. A scene is a bunch of objects, organized into layers. As you progress through the creative process, you use a set of window screen layouts specifically designed to help you work efficiently through the creative process: model the objects and create the props, clothe the actors and dress the set (assign materials), define the action (animation), render the video, and produce the movie. You can tailor these screen layouts, and create custom layouts, to match your working preferences.

Planning Your Timeline

Shots within a scene are accomplished by moving a camera and/or actors through the scene for a few seconds. Time in Blender is measured in frames, and typical video has 25 or 30 frames per second (fps), and film is 24 fps. For a five−second shot then, you allocate up to 150 frames for that shot (30 fps x 5 seconds). Giving yourself some wiggle−room, shot 2 would start at frame 250 and go from there. A one−minute movie set in a single scene for North America video broadcast (NTSC standard) would have a timeline that goes up to 1800 final frames, and may be laid out over the course of 2500 frames. This timeline allows for cutting out 700 frames, picking the best 1800 frames (30 fps x 60 seconds = 1800 frames) less transition time.

Multiple Cameras

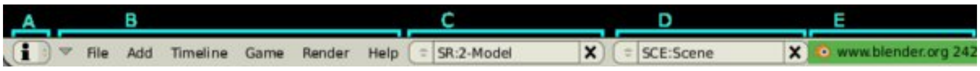
You can have multiple cameras in a scene, used for different shots, and select which one is active when rendering the shot.

Previous: Manual/Transform Properties	Contents	*Next:* Manual/Scene Creation

User Manual: Contents | Guidelines | Blender Version **2.43**

Current Screen Layout and Scene

Scenes are a way to organize your work. Scenes can share objects, but they can, for example, differ from each other in their rendered resolution or their camera view. The current window layout and scene is shown in the User Preferences window header, usually shown at the top of your screen:



User Information window header. **A**) Window type icon, **B**) Menu, **C**) Screen Layouts, **D**) Scenes, **E**) Version of Blender currently running (Click the Blender icon to the left to show splash screen)

Expandable Images in Manual:

Usually throughout this manual, if there is a little expanding icon in the bottom right−hand corner of the image, you may click it to see the image in a larger format. Also if when moving your mouse pointer over the image it changes shape to show the image can be clicked, this usually also means the image can be expanded/enlarged; Be aware that different wiki theme layouts can alter the location and appearance of the expand image indicator.

Loading the UI with File−>Open

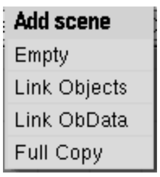
Inside each Blend file, Blender saves the user interface layout − the arrangement of screen layouts. By default, this saved UI is loaded, over−riding any user defaults or current screen layouts that you have. If you want to work on the blend file using your current defaults, start a fresh Blender, then open the file selector (F1). Turn off the "Load UI" button located in the file browser header, and then open the file. Blender will not disturb your current screen layout when it loads the new file.

Working with Scenes

Select a scene to work on by clicking on the up−down arrow next to the Scene name. Scenes and the objects they contain are generally specific to the project you are working on. However, they too can be saved in their current state to be re−used by pressing Ctrl U. They will then appear the next time Blender starts or when the user selects File−>New.

Blender comes with one default scene, which contains a camera, a lamp, and a box. How exciting.

Adding a Scene



Add scene popup menu

You can make a full copy of the current scene, start over with a blank slate, or create a scene that has links back to the current scene; objects will show up in the new scene, but will actually exist in the old one. Use this linking feature when, for example, the original scene contains the set, and the new scene is to contain the actors or props.

Starting Over

If you start with a new scene, be sure to add a camera and lights first.

Scenes are listed alphabetically in the drop−down list. If you want them to appear in a different order, start them with a numerical ordinal, like "1−". The internal reference for a scene is the three−letter abbreviation "SCE".

To add a scene, click on the scene list button, and select Add New. While you are adding a new scene, you have these options:

Empty
: Create a completely empty scene.

Link Objects
: All objects are linked to the new scene. The layer and selection flags of the Objects can be configured differently for each scene.

Link ObData
: Duplicates objects only. ObData linked to the objects, e.g. mesh and curve, are not duplicated.

Full Copy
: Everything is duplicated.

Usually, for your first scene, you make a full copy of the default. Alternatively, you can just start with the default, and start editing the cube that is usually hanging around waiting for you to do creative things. Get Blending!

Naming a Scene

By Shift LMB clicking on the Scene Name (usually Scene.001), you can change the name of the scene. For example, "BoyMeetsGirl" is usually the first of three acts.

You then proceed to model the props and objects in the scene using the 2−Model window layout.

Linking to a Scene

You can, at any moment, link any object from one scene to another. Just open the scene where these objects are, use Ctrl L > To Scene... and chose the scene where you want your objects to appear. Those will be linked to the original objects; to make them single user (independant, unlinked...) in a given scene go to that scene, select them and use U. You will be presented with a few options that allow you to free up the datablocks (Object, Material, Texture...) that you want.

Removing a scene from the file

You can delete the current scene by clicking the X next to the name.

Previous: Manual/Scene Management Contents *Next:* Manual/The Outliner

User Manual: Contents | Guidelines | Blender Version **2.43**

Outliner window

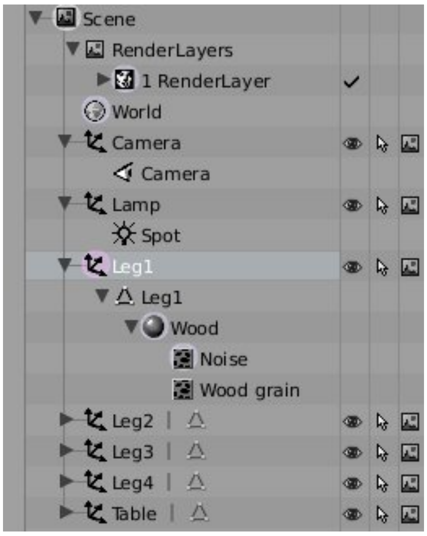
Description

The **Outliner Window** is used for easily navigating a complex scene. There are two views, the **Outliner** view and the **OOPS Schematic** view. The OOPS Schematic and Outliner give you a 2d representation of your complicated 3d world. Use these views to find things in your scene.

For example, suppose you sneeze while moving an object; your mouse flies off your desk (gezhundeit!) and the object is hurled somewhere off screen into space. Simply use the schematic/outliner to find it; select it, and move back to your 3d window to snap your cursor to it and then move it back.

Another more practical example is to evaluate the impact of a change on related datablocks. Suppose you are looking at your TableTop object, and it doesn't look right; the Wood material doesn't look right; you want it to look more like mahogany. Since the same material can be used by many meshes, you're not sure how many things will change color when you change the material. Using the OOPS Schematic, you could find that material and trace the links that it has to every mesh in your scene.

Outliner view



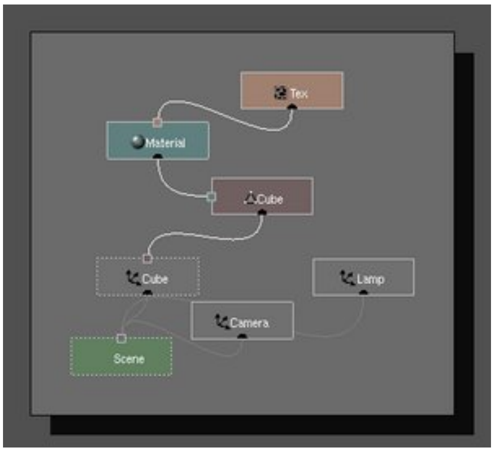
The Outliner window in list mode.

The **Outliner** is a kind of list that organizes related things to each other. In the Outliner, you can:

- View the data in the scene
- Select and deselect Objects in the scene
- Hide or show an object in the scene
- Enable or disable selection (to make an object "unselectable" in the 3D Window)
- Enable or disable the rendering of an object
- Select data like materials and textures directly (they show up automatically in the Buttons Window!)
- Delete objects from the scene

- Unlink data (equivalent to pressing the "X" button next to the name of a datablock)
- Easily select which RenderLayer to render
- Easily select which render pass to render (for example, you can choose to render just the Specular layer).

OOPS Schematic view



The OOPS schematic view in the Outliner window.

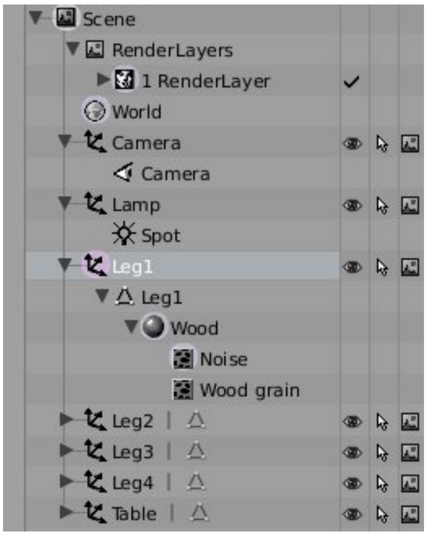
The **OOPS schematic** is a kind of picture that shows how things are linked together. **OOPS** is a highly geeky term for **Object−Oriented Programming System**. Yeah, right. I think someone just spilled coffee on their keyboard late one night and this was the first word that came to their mind ;) In the **OOPS** view, you can:

- Look at relationships *between* objects (for example, which objects use the same texture)

The main difference is that the OOPS schematic shows you *all available* things (datablocks) in your blend file, organized by type of thing: scenes at the bottom, objects in the middle, materials toward the top. The Outliner shows you things *in use* within your blend file, organized by parent object with their children as indents.

Selecting the Outliner Window Type

Turn a window into an Outliner window type by using the Window Type menu.



The Outliner window in list mode.

Choose a window and click on its current Window Type button (left−most icon in its header), and select Outliner.

Switch between the Outliner view and the OOPS Schematic view using the menu item View Show OOPS Schematic or View Show Outliner.

Window size

Choose or arrange a window size that suits the view you are going to work with. The OOPS Schematic needs a wide window, and the Outliner needs a tall, narrow window.

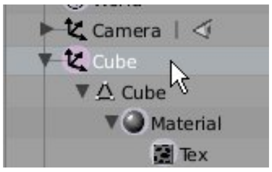
Using the Outliner view

Each row in the Outliner view shows a datablock. You can click the down−arrow to the left of a name to expand the current datablock and see what other datablocks it contains.

You can select datablocks in the Outliner, but this won't necessarily select the datablock in the scene. To select the datablock in the scene, you have to activate the datablock.

Selecting and activating

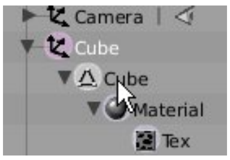
Single selection doesn't require any pre−selection: just work directly with LMB and/or RMB *inside* the name/icon area.



Toggling selection of a datablock.

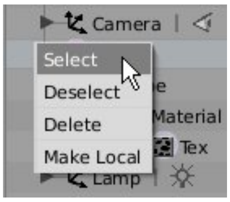
- **Toggle pre−selection of a group of datablock**
  Useful when you want to select/deselect a whole bunch of datablocks. For this you must prepare the selection using, to your liking:
  RMB or LMB ,
  Shift RMB or Shift LMB ,
  RMB and drag or LMB and drag
  all *outside* the name/icon area.
  You then confirm with a RMB *on* the name/icon area to bring on a dialog.

When you select an object in the list,| it is selected and becomes the active object in all other 3D View window panes. Use this feature to find objects in your 3D View; select them in the outliner, then snap and center your cursor on them via Shift S−>Cursor to Selection, and then C



Click LMB on the mesh data of the cube to activate Edit Mode.

- **Activate** the datablock with LMB on the *icon* or the *name* of the datablock. Activating the datablock will automatically switch to the relevant mode or Buttons context. For example, activating the mesh data of the cube will select the cube and enter Edit Mode (see right). Another example is that activating the material datablock for the cube will show the material in the Material context of the Buttons window.



Context menu for the Cube object.

- Show the **context menu** for a datablock with RMB  on the icon or name. Depending on the type of datablock, you will have the following options (*Note: some datablock types will not have a context menu at all*):
    - Select
    - Deselect
    - Delete
    - Unlink
    - Make Local
- **Delete** selected datablocks with X.
- **Expand** one level with NumPad +.
- **Collapse** one level with NumPad −.
- **Collapse/Expand all** levels with A

Toggling object−level restrictions

Icons for toggling

The following options are only available for objects:

- **Toggle visibility** by clicking the 'eye' icon for the object on the right−hand side of the Outliner. Useful for complex scenes when you don't want to assign the object to another layer. This will only work on visible layers − an object on an invisible layer will still be invisible regardless of what the Outliner says. V will toggle this property for any objects that are selected in the Outliner.
- **Toggle selectability** by clicking the 'arrow' icon. This is useful for if you have placed something in the scene and don't want to accidentally select it when working on something else. S will toggle this property for any objects that are selected in the Outliner.
- **Toggle rendering** by clicking the 'camera' icon. This will still keep the object

visibile in the scene. It will be ignored by the Renderer. R will toggle this property for any objects that are selected in the Outliner.

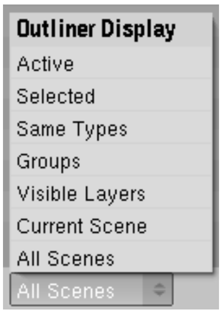
Searching

You can search the file for datablocks, either by using the Search menu in the header of the Outliner, or by using one of the following hotkeys:

- Shift F − Find again
- Ctrl Alt F − Find complete (case sensitive)
- Alt F − Find complete
- Ctrl F − Find (case sensitive)
- F − Find

Matching datablocks will be automatically selected.

Filtering the display

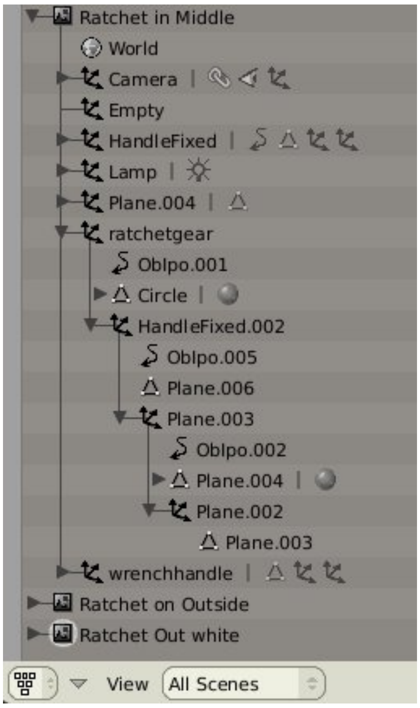


Outliner Display dropdown

The window header has a field to let you select what the outliner should show in the outline. By default, the outliner shows All Scenes. You can select to show only the current scene, datablocks that have been selected, objects that are on currently selected layers, etc. These selects are to help you *narrow the list* of objects so that you can find things quickly and easily.

- **All Scenes** − Shows *everything* the outliner can display (in all scenes, all layers, etc...)
- **Current Scene** − Shows everything in the current scene.
- **Visible Layers** − Shows everything on the visible (currently selected) layers in the current Scene. Use the Layers buttons to make objects on a layer visible in the 3D window.
- **Groups** − Lists only Groups and their members.
- **Same Types** − Lists only those objects in the current scene that are of the same types as those selected in the 3d window.
- **Selected** − Lists only the object(s) currently selected in the 3D window. You can select multiple objects by Shift RMB
- **Active** − Lists only the last selected object.

Example



The Outliner window in list mode.

The outline example shows that the blend file has three scenes: Ratchet in Middle, Ratchet on Outside, and Ratchet Out White. By clicking on the little arrow to the left of the name, the outline is expanded one level. This was done for the Ratchet in Middle scene. As you can see, this scene has some world material settings, a Camera, an Empty, a HandelFixed object ... all objects that were added to the scene.

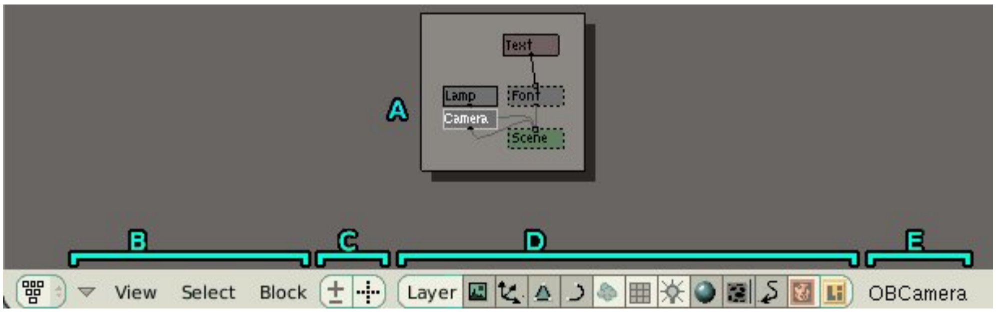
By clicking the arrow next to ratchetgear, we can see that it has some motion described by the ObIpo.001 curve; that it was based on a Circle mesh, and that it is the parent of HandleFixed.002. HandleFixed is in turn the parent of Plane.003, and so on.

The neat thing is: If you select any of these datablocks here, they will be selected in the 3d Window as well as far as this is possible. Pressing * on your keypad with your mouse cursor in any 3D Window will center the view to that object. Very handy. Also, pressing X will delete it, as well as all the other hotkeys that operate on the currently selected object.

Using the OOPS Schematic

Layout of the OOPS Schematic

In this view, the window has a clear background that, by default, shows the OOPS Schematic and a header:



Outliner Window in OOPS Schematic mode
A) OOPS Schematic, **B**) Menu, **C**) Zoom, **D**) Display Select, **E**) Currently selected datablock

The OOPS Schematic window & header have the following areas:

> **A**) The schematic picture
> **B**) Menus with the basic functions: View, Select, and Block
> **C**) A zoom control that allows you to focus on a certain area of the schematic.
> **D**) Visible Select − A number of buttons that toggle what kinds of datablocks are displayed in the schematic.
> **E**) The name of the currently selected datablock. The datablock is also highlighted in the OOPS schematic. (**A**)

Making sense of the OOPS Schematic

The schematic is a sort of map that shows the connections between datablocks. Each datablock is shown as a colored box. Boxes (datablocks) are connected by lines. Common types of connections between datablocks are:

Parents
: One datablock, let's say an object called "TableTop", is held up by four other objects "leg.001", leg.002", etc. The TableTop would be the parent of each of the legs, so that as the table top moves, the legs move as well. In the schematic, four lines would be shown going from the TableTop to each of the Legs.

Material Use
: Datablock can share the same material. In our Table example, the TableTop and each of the legs might share the same material, "Wood", so that they all look the same. In the schematic, there would be a box called "Wood" with five lines connecting it to each of the mesh datablocks TableTop, Leg.001, Leg.002, Leg.003 and Leg.004.

The schematic uses different colored boxes for each type of datablocks: green for scenes, grey for objects, taupe for text, sea green for materials, etc. to help you visually distinguish between types of datablocks.

The OOPS Schematic Header

View
: Handy functions include switching between the schematic and outliner view. Also, you can change the size of the boxes, so more can fit in the window.

Select
: Key functions include finding users and links between connected boxes, as indicated in the useage examples previously.

Block

Scales (S) the distance between multiple selected datablocks, and grabs/moves (G) an datablock or set of selected datablocks around the schematic − very useful for arranging and organizing your schematic.

Zoom controls

As you can imagine, depending on what you have selected and your scene complexity, these schematics can start looking like the piping diagram for a nuclear power plant. The schematic header provides two buttons to help you zoom in. Hold down LMB over the button and move your mouse up and down (forward and backward) to zoom your view in and out. Click on to start a border select. Select a region in the window, and your window view will be zoomed to that region.

Standard Window Controls

The window, like any Blender window, can be panned by clicking the middle mouse button while your cursor is in the window, and moving the mouse.

Visible Select

The series of icons in the header allow you to select what type(s) of datablocks are visible in the schematic. They are, left to right:

- ◊ Layer − Only show the datablocks from the shown layers.
- ◊ Scenes − Your stage, a set, where action occurs.
- ◊ Objects − Cameras, empties, and other misc items
- ◊ Meshes − The main things you model, not to be confused with Objects. e.g. One Mesh can be used in multiple objects and is displayed accordingly in the schematic.
- ◊ Curves, Surfaces, Fonts
- ◊ Metaballs − Mathematically calculated meshes that can mush together.
- ◊ Lattices − Deformation grids
- ◊ Lamps − All types of lights.
- ◊ Materials − Colors, paints.
- ◊ Textures − Color maps or gradients used commionly in materials and other places.
- ◊ Ipos − Actions,
- ◊ Images − Imported pictures
- ◊ Libraries − Collections of Objects

Datablocks

The base unit for any blender project is the datablock. The datablocks and the ways those are linked together are all there is to it, may it be a simple still image of a sphere floating over a plane or a full featured film. These datablocks can reside within as many files as needed for the good organization of the project.

Scene Datablock

Each "scene datablock" contains a scene. "Scene datablock" is the parent of the rest datablocks.

Object Datablock

Each "object datablock" has the properties of Scale, Location and Rotation and is the 'meeting place' for other datablock that define the other properties of that object when they are linked to it. An object can be linked to other datablock that determine its nature: mesh, curve, camera, lamp or armature datablock are a few

examples of such datablocks. Other datablocks, will define the material, the texture, the animation... for that object.

ObData Datablocks

These datablocks are such datablocks that are always connected to "Object datablock" in a way or another.

Curve Datablock

"Curve datablock" may contain NURBS curves or circles, Bezier curves or circles, or Text objects. It may also be linked to a "material datablock".

Camera Datablock

"Camera datablock" contains a camera.

Lattice Datablock

"Lattice datablock" contains a lattice.

Lamp Datablock

"Lamp datablock" contains a lamp.

Metaball Datablock

"Metaball datablock" contains a metaball.

Mesh Datablock

Each "mesh datablock" contains a mesh. "Mesh datablock" may contain link to one or more "material datablocks".

Material Datablock

"Material datablock" contains a material. It may contain links to "texture datablocks".

Note that "material datablocks" can be linked to "object datablock" instead if desired. You can set this in the "User Preferences" window below "Edit Methods".

Texture Datablock

"Texture datablock" contains a procedural or image texture.

Copying and Linking Datablocks

It is possible to copy object and scene datablocks.

Copying and Linking Scene Datablock

To copy scene datablock, use Scene list found in the header of "User Preferences" window. The list is to the right of the menus and window workspace list. Select "ADD NEW" to make a copy of the current scene. Select "Full Copy" from the list that opened to make a copy. The current scene will be copied to the new scene.

Instead of copying **everything**, you can link datablocks by selecting "Link Objects" or "Link ObData" on the list (the former copying the objects, but not their ObDatas − meshes, curves, materials, etc.). Note that if you select "Link Objects", it means that the objects are linked on deeper level as Object Datablock is parent of ObData datablock. So for instance if you move an object, the move is reflected to other scenes linked this way as well.

Copying and Linking Object Datablocks

- Shift D is used to make normal copy of the selected objects:
    - The object and some of it's child datablocks will really be duplicated, the other children are just linked; you can define the attributes to be duplicated in User Preferences Edit Methods, button group Duplicate with object:.

- Alt D makes a linked copy:
    - All datablocks but the object one are linked.

Copying and Linking other Datablocks

You can see a number next to the name of a datablock. This number indicated the number of links. If you click the number, it removes the link to the datablock and creates a new copy.

Previous: Manual/Scene Creation	Contents	*Next:* Manual/Using Linked Libraries

User Manual: Contents | Guidelines | Blender Version **2.46**

Linked Libraries Overview

Blender is able to "reach in" to other .blend files and pull in whatever you want. In this way, Blender supports reuse of your graphical models. For example, if you have a library .blend file that has a really neat *Material* used in it, you can, from your current .blend file, *Append* that *Material* into your current .blend file. This saves you from manually re−creating all the different settings.

General Procedure

Mode: All Modes

Hotkey: Shift F1

Menu: File Append or Link

The main menu in Blender is located in the User Preferences window (by default the header located at the top of your screen). From that menu, all you have to do is use File −> Append or Link or press Shift F1 in your active window. The active window will change to a File Browser (the Window type icon looks like a manilla folder) selector window. Use this window to navigate your hard drive and network−mapped drives through folders and subfolders to find the .blend file that has the object you want to reuse. When you click on a .blend file (indicated by the square box next to the name), Blender will go into that file and show you the list of datablock types within it: Scenes, Objects, Materials, Textures, Meshes, etc. Clicking on any one of them will display the specific instances of that type.

Folder and File Organization

We suggest creating a folder called /lib or /library. Under that library, create a set of folders for each kind of thing you might want to access and re−use later on, such as Materials, Textures and Meshes. Create subfolders under each of those as your library grows. For example, under the Meshes folder, you might want to create folders for People, Spaceships, Furniture, Buildings, etc. Then, when you have a .blend file that contains a chair mesh, for example, all you have to do is copy that file into the Furniture folder.

Appending library objects into your current project

The following procedure appends an object with all its linked data, such as mesh data, materials, texture etc., to the current .blend file.

- Select File −> Append or Link
- Locate and select the file that contains the object you want to append (often a 'library' file).
- Navigate to the OBJECT section of the file
- Select one object from the list using LMB, multiple objects via RMB, and/or a range of objects by dragging RMB
- Repeat the above for each kind of object you wish to append or link. Parents and Armatures (all modifier objects) must be selected separately.
- Set desired options that are shown in the header (to cursor, to active layer)
- LMB on Load Library or press Enter or MMB directly on the data to append

Of course, you can append or link many other things besides objects: cameras, curves, groups, lamps, materials, meshes, **an entire scene**, etc. Note that there is a BIG difference between adding the Object and the type of object, such as Mesh. If you append a Mesh datablock, you are only bringing in the data about that particular type of Mesh, and not and actual instance of the Mesh that you can see.

Use Append (button enabled by default) if you want to make a local independent copy of the object inside your file. Select Link if you want a dynamic link made to the source file; if anyone changes the object in the source file, your current file will be updated the next time you open it. These buttons are located in the File Browser window header.

Click Load Library to append or link the object into your current blend file.

Some more loading option buttons (in the File Browser header) include:

- AutoSel:
    - When an object is loaded, it is not active or selected; it just plops into your .blend file. Often, right after loading, you will want to do something with it, like scale it or move it. Enable this button and the imported object will be selected, just as if you magically right−clicked on it. This button saves the step of finding the object and selecting it.
- Active Layer:
    - Blender has 20 layers to divide up a large scene, and each object resides on some layer. By default, an object is loaded into your file directly into the layer it resides on in the source file. To load the object to the current active layer that you are working on, enable this button.
- At Cursor:
    - By default, an object is loaded into your file at the location it is at in the source file. To reposition the object to your cursor when it loads, enable this button.

Finding What was Loaded:

If the loaded object is not visible, consider using At Cursor or AutoSel. If you use AutoSel, remember there are Snap tools to put your cursor on the object (Shift S4 (Cursor to Selection)), and Center your view on it (C (Center View to Cursor)). Note that these tools do not work if the object is on an unselected layer, since objects on unselected Layers are invisible.

Reusing Objects (Meshes, Curves, Cameras, Lights, etc)

Let's suppose you created a wheel in one .blend file and want to reuse it for your current project. The physical model of the wheel would be a mesh, and probably comprised of a tire and rim. Hopefully you named this mesh something reasonable, like, oh, I don't know, "Wheel". The wheel may be colored and thus have some Materials assigned to it (like rubber and chrome).

Once you navigate to the file, select the "Wheel" and it will be imported into your current file. You can import a copy of it, or merely link to it.

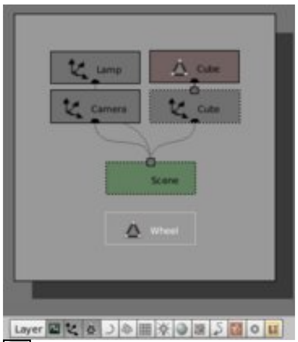
Linking: If you link to it, and later modify it in the source file, it will be shown "as−is" (modified) in your current file the next time you open it up.

Other artists have released their models to the public domain, and friends may share models simply by posting or emailing their .blend files to each other. Keeping these files, as well as your past projects, in a Download directory on your PC/server will save you from ever having to reinvent the wheel.

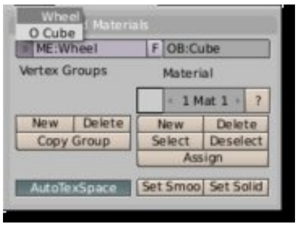
When selected, linked objects are outlined in Cyan. Normal selected objects are outlined in pink.

Notice that you cannot move a linked object! It resides at the same position it has in the source file. To move/scale/rotate the object, turn it into a Proxy.

Using Appended/Linked Mesh Data:



When Appending or Linking certain resources such as mesh data, it may not be instantly visible in the 3D Viewport. This is because the data has been loaded into Blender but has not been assigned to an Object, which would allow it to be seen. You can verify this by looking in the Outliner View and switching it to OOPS Schematic view (you may need to have the Displays Scene datablock button selected in the OOPS Schematic Header menu). In the OOPS Schematic picture you can see that Wheel is not linked to an Object.



To allow the newly loaded Wheel mesh to be assigned to an Object, either select a currently visible object or create a new object (such as a cube), then goto the Link and Materials panel and select the Wheel mesh from the mesh drop down panel, at that point you should see the Wheel mesh, because it's been assigned to an object.

If instead of Appending/Linking to a mesh you instead load the object into Blender, it should be instantly displayed in the 3D Viewport without having to associate an object with the mesh using the Link and Materials panel.

Reusing Material/Texture Settings

Some materials, like glass or chrome, can be very tricky to get "just right". The Blender Foundation has released, for example, a Materials CD, which is available for free to download from their site. Using the .blend files on that CD, you can import common materials, like glass, chrome, wood and bananas. This feature saves you a lot of time, as it often means you don't have to be fiddling with all the little buttons and sliders just to re−create a material. I call out the Banana material because it is a great example of using simple procedural materials with a ColorRamp, and a procedural texture, to give a very realistic look. When you navigate to the file, and select Materials, the browser will show you a sphere sample of that material to help you visualize the texture that goes with the name. For more information on using the image browser, see the release notes.

Blender Extension: Library

There is also a fantasic Python script called Blender Library that overarches all of your files and allows you to construct a master library. This script displays a preview and helps you organize your Blender work. Highly recommended; search www.blendernation.com for "Blender Library", it is also stored on the Blender Wiki Scripts section here.

Reusing Node Layouts

To reuse noodles (node layouts), open the original (source) file and create a Group for the set of nodes that you think you want to reuse. When you want to import that node group into your current file, select File−>Append from the User Preferences window header, and navigate to the file. When you dive into the file, there will be a NodeTree option. Click it and the list of node groups in that file will be listed. LMB Click the one you want and then Load Library.

[Verse]

Verse is an amazing OpenSource collaboration tool that integrates with Blender. Verse enables multiple people to work on, link, and share objects and modifications in Blender files in real time.

Proxy Objects

A proxy is a legal stand−in or substitute for the real thing. In Blender, when you make a linked copy (described above), you cannot edit the object; all you have is a link to it. You cannot add to it or change it, because its source is in another file that is not open.

When working in a team environment, you may want more flexibility. For example, if modeling a car, you may have one person working on the shape of the car (its Mesh), but another working on available color schemes (its Materials). In this case, you want to grant the Painter a Proxy of the object and allow him/her to modify the material settings. More commonly, you will have a character being animated by a team of animators; they can define poses, but cannot change the character's colors or armature, only use what is defined by the master rigger.

The important aspect of a Proxy Object is that it allows you to edit data locally, but also allows specific data to be kept protected. Data that's defined as protected will always be restored from the Library (typically on file reading or undo/redo steps). This protection is defined in the referenced Library itself, which means that only the Library files can define what's allowed to change locally.

For Poses, you can control this by indicating Bone layers as being protected. A protected layer is shown with a black dot in it. Use CTRL+click on a button to protect or unprotect that layer.

Mode: Object Mode

Hotkey: Ctrl Alt P

To make a Proxy object for yourself, establish a Link to the source object as described above. With that linked copy selected (RMB and in view (you can see it in the 3D View), press Ctrl Alt P and confirm the Make Proxy dialog. The object will be named with the original name plus a "_proxy" suffix. You may now move and modify the proxy. When selected, it will look like a local object (outlined in pink).

You can then edit unprotected data. For most objects, this includes the location and rotation. You can also animate the object's location and animation using Ipo Curves. For mesh objects, the shape of the mesh is protected, so you cannot define shape keys. When you reload your file, Blender will refresh your file with any changes made to the original protected data, but will not reset your changes (unless the owner has).

Previous: Manual/The Outliner	Contents	*Next:* Manual/Modelling

User Manual: Contents | Guidelines | Blender Version **2.43**

Modelling in Blender

As you have seen in the Quick Start chapter, the creation of a 3D scene needs at least three key things: Models, Materials and Lights. In this Part we will delve deeper into the first of these issues Modelling. Modelling is the art and science of creating a surface that mimics the shape of a real−world object or fits your imagination of abstract objects.

Objects come in many forms, shapes and sizes, so Blender has many different tools available to help you make your model quickly and efficiently:

- Objects

  Working with objects as a whole
- Meshes

  Working with the mesh that defines the shape of an object
- Curves

  Using Curves to model and control objects
- Surfaces

  Modeling a NURBS surface
- Text

  Textual tools for putting words in 3D space
- Meta Objects

  Globs and Globules
- Dupliverts and Frames

  Duplicating Meshes
- Modelling Scripts

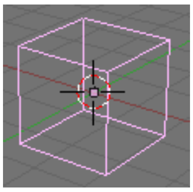
  Since Blender functionality is extensible via Python, there are a number of very useful scripts that assist you in modelling.

Many people use "box modelling" which starts with a basic cube, and proceeds with extruding and moving vertices to create a larger, more complicated mesh. For flat objects, like walls and table tops, you can use "curve modelling" which defines the outline using bezier or Nurbs curves, and then extrudes it to the desired thickness. Either method is fully supported in Blender using its modelling tools.

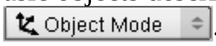
Previous: Manual/Using Linked Libraries Contents *Next:* Manual/Objects

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.4**

Object Mode



Selected object.

The geometry of a scene is constructed from one or more Objects. For example Lamps, Curves, Surfaces, Cameras, Meshes, and the basic objects described in [Basic Mesh Objects](#). Each object can be moved, rotated and scaled in Object Mode, . For performing more detailed changes to the geometry, you can use [Edit Mode](#).

Once you've added a basic object (see [Basic Mesh Modelling](#)), you are automatically switched into [Edit Mode](#) if the Object is a Mesh, a Curve or a Surface. You can switch back to Object Mode by pressing TAB. The object's wireframe, if any, should now appear pink, meaning that the object is now selected and active; see (*Selected object.*)

Erase

Mode: Edit or Object mode

Hotkey: X or DEL

Menu: Object â†’ Delete

Description

Erases or deletes selected objects.

Join

Mode: Object mode

Hotkey: Ctrl J

Menu: Object â†’ Join Objects

Description

Joins all selected objects to one single object. (The objects must be of the same type.) The center point of the resulting object is obtained from the previously *active* object. Performing a join is equivalent to adding new objects while in Edit mode.

Select Links

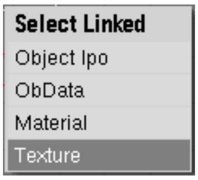
Mode: Object mode

Hotkey: Shift L

Menu: Select â†’ Select Linked

Description

Select all objects sharing a link with the active one. You can select objects sharing an IPO, data, material, or texture link (*Selecting links.*).



Selecting links.

- Object Ipo

  Selects object that share IPO information.

- ObData

  Selects object that share *data* information.

- Material

  Selects object that share Material information.

- Texture

  Selects object that share Texture information.

Previous: Manual/Modelling | Contents | *Next:* Manual/Selecting Objects

User Manual: Contents | Guidelines | Blender Version **2.45**

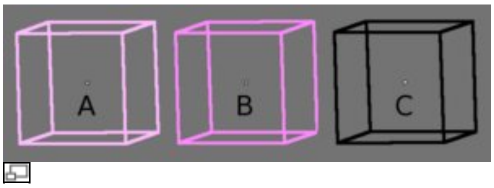
Introduction

Selection, in almost any program, determines which elements will be the target of our actions. As such, the more adapted the selection tool is to the action intended the better. Tools and functions are in a great number in Blender and so are it's selection methods.

What follows is a short description of the concepts and selection tools which are available in Object Mode.

Selections and the Active Object

Blender distinguishes between two different states of selection:

- 

  A) Selected Active Object, B) Selected Object, C) Unselected Object. Outlines have been thickened to make them easier to distinguish.

  In Object Mode the last selected item is called the "Active Object" and is outlined in pink (the others are purple). There is exactly one Active Object at any time (unless nothing is selected).

Many actions in Blender use the Active Object as a reference, for example the boolean tools or linking operations. If you already have a selection and need to make a different object the active one, simply re−select it with Shift RMB .

- All other selected objects are just that, selected. You can select any number of objects.

Point Selection

The simplest form of object *selection* consist into using RMB on it.

To *add to the selection* we use Shift RMB on more objects.

If the *objects are overlapping* in the view we can use Alt RMB to get a list of possible choices.

If we want *to add to a selection* this way then the shortcut becomes ShiftAlt RMB .

Activation of an object that is already selected is done with a Shift RMB click on it.

Deselection is achieved by one Shift RMB on an active object and two such clicks if the object wasn't active.

Rectangular or Border Select

Mode: Object mode

Hotkey: B

Menu: Select â†’ Border Select

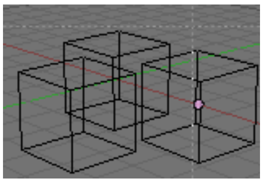
Description

With Border Select we draw a rectangle while holding down LMB. Any object that lies even partially within this rectangle becomes selected.

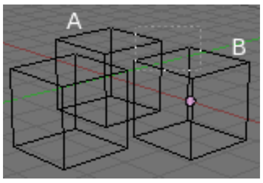
For deselecting objects we use either of MMB or RMB.

Example

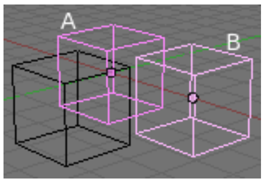
In (*Start*) Border Select has been activated and is indicated by showing a dotted cross−hair cursor. In (*Selecting*), the *selection region* is being chosen by drawing a rectangle with the LMB. The rectangle is only covering cubes "**A**" and "**B**". Finally, by releasing LMB the selection is complete; see (*Complete*).



Start.



Selecting.



Complete.

Notice in (*Complete*) the bright color of selected cube "**B**". This means it is the "Active Object", the last selected object prior to using the Border Select tool.

Hints

Border select adds to the previous selection, so in order to select only the contents of the rectangle, deselect all with A first.

Lasso Select

Mode: Object mode

Hotkey: CTRL+ LMB

Menu: no entry in the menu

Description

Lasso select is used by drawing a dotted line around the pivot point of the objects, in ObjectMode.

Usage

While holding CTRL down, we simply have to draw around the pivot point of each object we want to select with LMB.

Lasso select adds to the previous selection. For deselection we use Shift Ctrl LMB.

Menu Selection

The selection methods described above are the most common. There are also many more options accessible through the 'Select' menu of the 3D view or the 'Select' option of the SpaceBar menu.
Each is more adapted to certain operations.

Select Grouped

Mode: Object mode

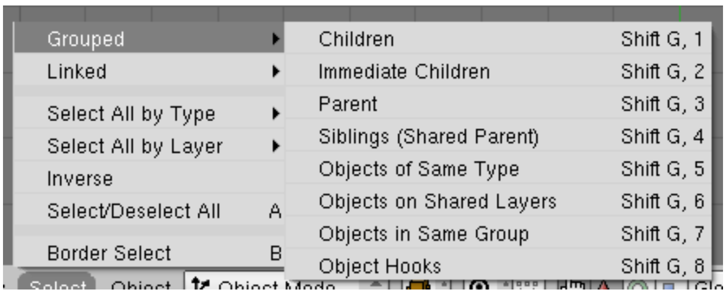
Hotkey: Shift G

Menu: Select â†’ Grouped

Description

There are two ways to organize the objects in relation to one another. The first one is parenting, and the second simple grouping.

We can take advantage of those relationships to select members of those families or of those groups.



Options for parented and grouped objects.

Options

Select Grouped in Object Mode uses the active object as a basis to select all others.

Available options are:

- Children

  Selects all children of the active object recursively.

- Immediate Children

  Selects all direct children of the active object.

- Parent

  Selects the parent of this object if it has one.

- Siblings (Shared Parents)

  Select objects that have the same parent as the active object. This can also be used to select all root level objects (objects with no parents).

- Objects of Same Type

  Select objects that are the same type as the active.

- Objects on Shared Layers

  Objects that have at least 1 shared layer.

- Objects in Same Group

  Objects that are part of a group (rendered green with the default theme) will be selected if they are in one of the groups that the active object is in.

- Object Hooks

  Every hook that belongs to the active object.

Select linked

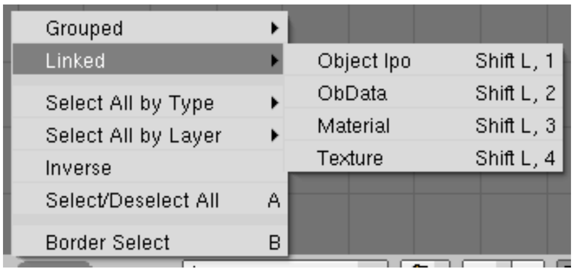
Mode: Object mode

Hotkey: Shift L

Menu: Select â†’ Linked

Description

Selects all objects which share a common datablock with the active object.



Options for objects which share a datablock.

Options

Select Linked in Object Mode uses the active object as a basis to select all others.

Available options are:

- Object Ipo

  Selects every object that is linked to the same Ipo datablock of the Object type. Any other type like Constraint, Pose, won't work.

- ObData

  Selects every object that is linked to the same ObData, i.e. the datablock that specifies the type (mesh, curve, etc.) and the built (constitutive elements like vertices, control vertices, and where they are in space) of the object.

- Material

  Selects every object that linked to the same material datablock.

- Texture

  Selects every object that linked to the same texture datablock.

Select All by Type

Mode: Object mode

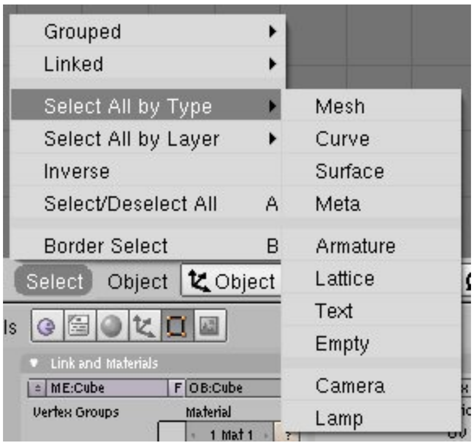
Hotkey: ???

Menu: Select â†’ Select All by Type

Description

The types are Mesh, Curve, Surface, Meta, Armature, Lattice, Text, Empty, Camera, Lamp.

With this tool it becomes possible to select every **visible** object of a certain type in one go.



Options for objects of one type.

Options

Select All by Type in Object Mode offers an option for every type of object that can be described by the ObData datablock.

Just take your pick.

Select All by Layer

Mode: Object mode

Hotkey: ???

Menu: Select â†’ Select All by Layer

Description

Layers are another means to regroup our objects to suit our purpose.

This option allows the selection of every single object that belongs to a given layer, visible or not, in one single command. This selection is added to anything that was already selected at that moment.

Choice of one layer.

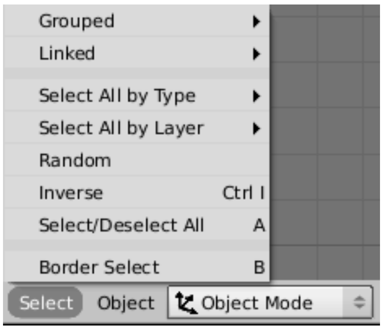
Options

We have the option of selecting the objects of one single layer at a time by LMB on it's number. This has to be repeated for each new layer.

Selection of Objects:

Rather than using the Select All by Layer option, it might be more efficient to make the needed layers visible and use A on them. This method also allows objects to be deselected.

Other Menu Options



Options for parented and grouped objects.

Available options on the first level of the menu are:

- Random

  Randomly selects unselected objects based on percentage probability on currently active layers. On selecting the command a numerical selection box is displayed for the user to select the percentage chance that an object will be selected.

  Random Select Percentage.

  It's important to note that the percentage represents the likelyhood of an unselected object being selected and not the percentage amount of objects that will be selected.

- Inverse

  Selects all objects that were not selected while deselecting all those which were.

- Select/Deselect All

  If anything was selected it is first deselected. Otherwise it toggles between selecting and deselecting every visible object.

- Border Select

  As described above in the section on border select.

Previous: Manual/Objects | Contents | *Next:* Manual/Editing Objects

User Manual: Contents | Guidelines | Blender Version **2.4x**

Moving (translating) objects

There are two ways to move or translate an object: moving it by itself, or moving it relative to something else.

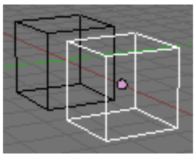
Moving Object(s) Individually

Mode: Object mode

Hotkey: G or Gesture

Menu: Object â†’ Transform â†’ Grab/Move (or Grab/Move on Axis for constraints)

Description



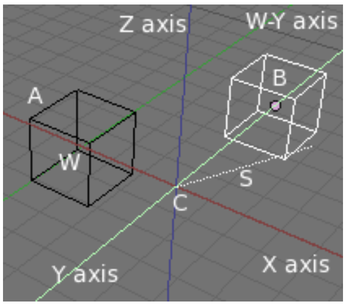
Grab mode.

To translate an object is to place an object in Grab mode. The selected objects will be displayed as white wireframes and can be moved with the mouse (without pressing any mouse buttons); see (*Grab mode*) or keyboard arrow keys.

To confirm the new position, click LMB or press ENTER; to cancel Grab mode, click RMB or press ESC. The header of the 3D Window displays the distance you are moving.

Options

Axis constraint



Global axes.

Note

For this section, and it sub sections, please reference (*Global axes*).
Movement can be constrained to an axis that is aligned with one of the axes of the global coordinate system, centered on the object's original world location. The cube "**B**"'s original world location is labeled "**C**". The center of the global coordinate system is labeled "**W**"; the Z axis is not visible.

By constraining movement to a global axis you are, in effect, restricting movement to one dimension.

The global aligned axes are color coded as follows:

- X axis is dark red. Labeled "**X axis**".
- Y axis is dark green. Labeled "**W−Y axis**".
- Z axis is dark blue. Labeled "**Z axis**".

The restricted axis is always highlighted in a lighter shade of color. For example, the Y axis is drawn in light green if movement is constrained to the Y axis; labeled "**Y axis**".

There are two ways to constrain movement: using the mouse or using the keyboard.

Using the mouse

To lock or constrain movement using the mouse, enter Grab mode and move the object while pressing MMB. While in Grab mode you can use the Gesture System to pre−select an axis by moving the mouse in a direction roughly inline with a world axis and then clicking and releasing MMB. For example, if you move the mouse along what visually appears to be the X axis and then click and release MMB the object's movement will be restricted to the world X axis.

Alternately, you can interactively choose the constraining axis by dragging with the MMB while in Grab mode. All three axes become visible with a guide line that emanates from the object's original location; labeled "**C**". This guide is drawn in white dotted line labeled "**S**". As the guide line nears an axis that axis becomes highlighted in a lighter shade and the object snaps to that axis. In this example the guide line is near the Y axis and the cube, labeled "**B**", snaps to it.

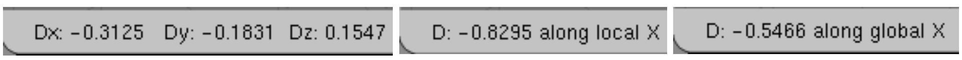
If you keep CTRL pressed while moving the object you will activate Snap mode, and the object will move by a whole number of units (grid squares). Snap mode ends when you release CTRL so be sure to confirm the position before releasing it. For finer snapping you can hold both CTRL and SHIFT.

You can control positioning to a finer degree by holding SHIFT while you move. Large mouse movements will translate into very small object movements, which allows for finer positioning.

The location of selected objects can be reset to the default value by pressing Alt G. The default location is the origin of the global coordinate system.

Using the keyboard

You can constrain movement to a given axis by pressing either X, Y or Z. A single key press constrains movement to the corresponding *global* axis (*Global Constraint*), as MMB does. A second keypress of the *same* key constrains movement to the corresponding Object *local* axis (*Local Constraint*) and a third keypress of the same key removes constraints, (*No Constraint*). The constrained axis is drawn in a lighter color to better visualize the constraint. (*Local Constraint*) and (*Global Constraint*) are all examples of constraints on the X axis using the X key.



No Constraint. Local Constraint. Global Constraint.

Once grabbing is activated you can enter the Object translation manually by simply typing in a number. This will change the 3D window header as shown in (*Manual entry*).

Manual entry.

The number entered is a distance number (i.e. how far from the object's current location). Think of the "**D**" as in displacement, delta or distance. The number entered is *not* a world coordinate. To change the object's world coordinates see Transform Properties Panel.

By default the X component field is where entry initially goes; see field labelled "**Dx**" in (*Manual entry*). You can change the default by using the TAB prior to entering any numbers. For example, to translate 4.4 units along the Y axis you would:

- Enter Grab mode.
- TAB once.
- Type 4.4.

To translate 3.14 units on the Z axis you would use the TAB key *twice* prior to entering the numbers.

Currently you can't delete an incorrect number. You must restart by returning to the original numbers. The BACKSPACE key will reset to the original values. Hit ENTER or SPACE to finalize and ESC to exit. If you want more flexibility with manual entry see Transform Properties Panel.

It is also possible to enter a value followed by an axis letter to indicate that the value that is entered should be made along the specified axis letter. For example if you wanted to move an object along the y axis by 3 Blender units you would type Gy3Enter or G3yEnter. You can also enter negative values to move in the opposite direction.

3 Axis Coordinate/Displacement Entry using TAB:

In entering X/Y/Z axis numbers at the keyboard, you can use the TAB to cycle through the X/Y/Z fields that will be altered when a number is entered from the keyboard. It is important to realize that in cycling through the fields you can fill in each field with a different value as you cycle through them, you do not just have to fill in only one field. For example if you wanted to move an object by X 2 Y 3 Z 4, you would type G2 TAB3 TAB4 ENTER

As well as being able to constrain along a single specified axis, it is also possible to prevent axis translation/scaling along one axis, but allow translation/scaling along the other two axis; This is achieved by pressing either Shift X, Shift Y or Shift Z to prevent translation/scaling along the specified axis. So if you wished to scale an object on the X and Z axis's but not the Y axis you could type S Shift Y.

Hints

You can use the keyboard's "." and the numeric keypad's "." for decimals entry. Be aware that older versions of Blender may not allow the use of the numeric keypad's "." for entering decimals.

Moving/Translating Object(s) by Changing Attributes

Mode: Object mode

Hotkey: Ctrl C

Menu: Object â†’ Copy Attributes â†’ (select a set)

Description

Blender has a general purpose way of copying any active object's attributes to any number of other selected objects. If you copy one object's location attribute to another object, that second object will be "moved".

Options

The attributes that can be copied include:

- Location
- Rotation
- Size
- Drawtype
- Time Offset
- Dupli
- Mass
- Damping
- Properties
- Logic Bricks
- Protected Transform
- Object Constraints
- NLA Strips
- Texture Space
- Subsurf Settings
- Modifiers
- Object Pass Index

So, if you shift−select a cube and a cone, and then shift−select a lamp last (the lamp thus being the *active* object), and choose Ctrl C 1, the cube and the cone will be "moved" to the same location as the lamp.

Rotating objects

There are two ways of changing an object's rotation; individually, and by copying the rotation attribute from another object as described above.

Mode: Object mode

Hotkey: R or Gestures

Menu: Object â†’ Transform â†’ Rotate / Rotate on Axis

Description

Change the rotation by moving the mouse and confirming with LMB or ENTER. You can cancel with RMB or ESC.

Rotation in 3D space occurs around an axis, and there are several ways to define this axis. But in general an axis is defined by a direction line and a point that the line passes through. By default the axis is orthogonal to your screen (i.e. it is going into or out of your screen). If you are viewing the scene from the front, side, or top 3D view windows, the rotation axis will be parallel to one of the global coordinate system axes. If you are viewing the scene from an angle, the rotation axis is angled too, which can easily lead to a very odd rotation of your object. In this case, you may want to keep the rotation axis parallel to the coordinate system axes.

Examples

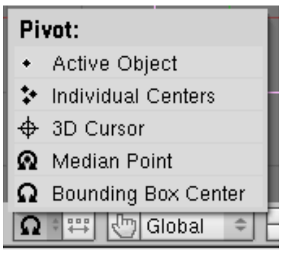
As you rotate the object the angle of rotation is displayed in the 3D window header:

Options

Axis of rotation Constraint

Just like Grab mode you can constrain the axis of rotation by using either the mouse or the keyboard. The only difference is that you only enter an angle. See Grab mode's Axis constraint for exact details.

Point of rotation



Pivot menu

To select the point−of−rotation that the rotation axis will pass through, use the Rotation/Scaling button accessed in the header of the 3D window, . This will display the (*Pivot menu*).

Active Object
: The axis passes through the active object (drawn in pink). See Selecting objects.

Individual Object Centers
: Each selected object receives its own rotation axis, all mutually parallel and passing through the center point of each object, respectively. If you select only one object, you will get the same effect as with the *Bounding Box Center* button. You can also select this by presssing Ctrl ..

3D Cursor

The axis passes through the 3D cursor. The cursor can be placed anywhere you wish before rotating. You can use this option to easily perform certain translations at the same time that you rotate an object. You can also select this by presssing ..

Median Point
The axis passes through the median point of the selection. This difference is only relevant in Edit Mode, and the *Median* point is the barycentrum of all vertices. You can also select this by presssing Ctrl ,.

Bounding Box Center
The axis passes through the center of the selection's bounding box. If only one object is selected, the point used is the center point of the object, which might not necessarily be in the geometric center. You can also select this by presssing ,.

For finer control or precision use CTRL or SHIFT. Pressing CTRL switches to *Snap mode* and rotations are constrained to 5 degree increments. Pressing SHIFT at the same time contraints the rotation to 1 degree increments. Pressing SHIFT alone while rotating allows finer degrees of rotation as precise as 1/100th of a degree. The rotation of selected objects can be reset to the default value by pressing Alt R.

If you're just getting started with rotation, don't worry too much about the foregoing details. Just play around with the tool and you'll get a feeling for how pivot points effect rotation. For example, an easy way to understand how pivot points work is to create two cubes as shown in (*Multiple Selected*). Then cycle through each pivot point type while in Rotate mode.

Hints

To have one cube orbit another cube set the pivot point to *Active Object*. As you rotate, constrained or not, the other object(s) orbit the active object.

Scaling objects

Mode: Edit mode / Object mode

Hotkey: S or Gesture System

Menu: Mesh â†’ Transform â†’ Scale

Description

Scale the objects by moving the mouse and confirming with LMB or ENTER, and cancel with RMB or ESC.

Scaling in 3D space occurs around a center point; much like a rotation occurs around a pivot point. If you increase the size of the object, all points are moved away from the selected center point; if you decrease it, all points move towards this point.

Options

Axis of scale Constraint

By default, the selected objects are uniformly scaled in all directions. To change the proportions (make the object longer, broader and so on), you can lock the scaling process to one of the global coordinate axes, just as you would with Grab mode and Rotate mode. Again all considerations on constraining to a specific axis, in respect to Grabbing, still hold as well as those on numerical input. See Grab mode's Axis constraint for exact details.

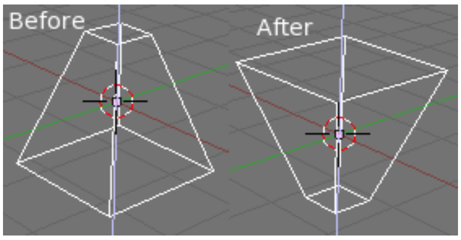
Center point of scale

To select the center−point−of−scale use the Rotation/Scaling button accessed in the header of the 3D window, . This will display the (*Pivot menu*) as shown in Point of rotation.

Here again the CTRL key switches to Snap mode, with discrete scaling at **0.1** steps. Press SHIFT for fine tuning. The scaling of selected objects can be reset to the default value by pressing Alt S.

Mirroring objects

Mirroring objects is a different application of the scale tool. Mirroring is effectively nothing but scaling with a negative factor in one direction. For example, to mirror in the direction of any single axis:



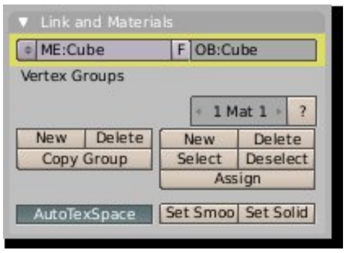
Mirrored Frustum

- Enter Scale mode
- Select an axis using X, Y or Z key.
- Enter '−1' as the scaling factor.

(*Mirrored Frustum*) is an example of mirroring a frustum object along the Z axis. These are the steps to mirror the frustum:

- Enter Scale mode
- Select the Z axis using the Z key.
- Enter '−1' as the scaling factor.
- Hit ENTER

Skinning and Cloning Objects



Relevant fields highlighted in yellow.

At the very top of the Links and Materials Panel, you will find two fields, one in light pink and another right next to it in gray.

The field in gray starts with OB: and is the name of the object itself. It has to be unique within the .blend file across all scenes. The field name on the left starts with a two−letter abbreviation indicating what type of object it is, and the name of its skin, or physical appearance:

- ME: Is the physical mesh, made up of vertices.
- CU: Is a curve, surface, or text object, made up of control points.
- MB: Is a metaball, whose skin is represented as a mathematical function.

Any of these skins can be shared by objects. Imagine a scene with 50 cats, some skinny, some fat. You would have two meshes, ME:Cat.Skinny and ME:Cat.Fat. You would create 50 OB:Cat.001, OB:Cat.002, ... OB:Cat.050 and assign 20 of the OB to be fat cats, and the rest skinny.

Options

Clicking the F will fake a user of the skin, and it will not be deleted when no one uses it. The next time you open the .blend file, it will be in memory and will not have to be re−made. You can then create an object of its type, and use that skin.

At any time you can change the skin of an object by clicking the up−down selector on the left of the field and selecting a different skin for that same object type. When you do, the field will then show the multi−user button, "2" identifying how many other objects share this skin.

Hotkey: Alt D

Menu: Object −> Duplicate Linked

Select an object and use the hotkey to create a clone of the original. The two objects will share the same skin. This means that altering either object at the Edit Mode level (when in Edit Mode), by for example grabbing vertices, will result in the other objects being altered in the same relative way. This linking usually only works when in Edit Mode, so scaling/rotating/grabbing an object in Object Mode will not result in the other linked object being affected.

Complex Objects

To change an object's shape, you edit it by selecting it and pressing Tab. You then choose a selection mode (vertices, edges, or faces), select them and then grab, rotate, or scale them. These operations are the same as described above for whole objects.

For working on complex models, hide the parts you are not working on (select in edit mode and press H to hide, Alt H to unhide). If you set up vertex groups for the different parts of your model, it is easy to select parts to hide. The other option is to separate your complex thing into multiple parts, and then only edit one part at a time. You can also hide objects in your scene so they don't clutter the view. You can also move objects to layers, and then not select those layers so they are not shown in the 3D view.

Previous: Manual/Selecting Objects	Contents	*Next:* Manual/Groups and Parenting

User Manual: Contents | Guidelines | Blender Version **2.42**

There can be many objects in a scene: A typical stage scene consists of furniture, props, lights, and backdrops. Blender helps you keep everything organized by allowing you to group like objects together.

When modelling a complex object, such as a watch, you may choose to model the different parts as separate objects. However, all of the parts may be attached to each other. In these cases, you want to designate one object as the parent of all the children. Movement and rotation of the parent also affects the children.

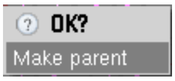
Parenting objects

Mode: Object mode

Hotkey: Ctrl P

Menu: Object â†’ Parent â†’ Make Parent

Description



Make Parent

To parent objects, select at least two objects, and press Ctrl P. A confirmation dialog will pop up asking Make Parent. Selecting Make Parent confirms and the child/children to parent relationship is created, see image (*Make Parent*). The last object selected will be the Active Object outlined in pink, and will also be the Parent Object. If you selected multiple objects before selecting the parent, they will all be children of the parent and will be at the same level of the hierarchy.

Moving and rotating the parent will also usually move/rotate the child/children. However moving/rotating the child/children of the parent, will not result in the parent moving/rotating. The direction of influence is usually from Parent to Child/Children, not from Child/Children to Parent.

Mode: Edit mode

Hotkey: Ctrl P

Menu: Mesh â†’ Vertices â†’ Make Vertex Parent

You can parent an object to a single vertex or a group of vertices as well; that way the child/children will move when the parent mesh is deformed, like a mosquito on a pulsing artery. In Object Mode, select the child/children and then the Parent Object. Tab into Edit Mode and on the Parent Object select either 1 vertex that defines a single point, or select 3 vertices that define an area (the 3 vertices do not have to form a complete face they can be any 3 vertices of the Parent Object), and then Ctrl P and confirm. At this point if a single vertex was selected a relationship/parenting line will be drawn from the vertex to the Parent Object. If 3 vertices are selected then a relationship/parenting line is drawn from the averaged center of the 3 points (of the Parent Object) to the child/children. Now, as the parent mesh deforms and the chosen parent vertex/vertices move, the child/children will move as well.

Options

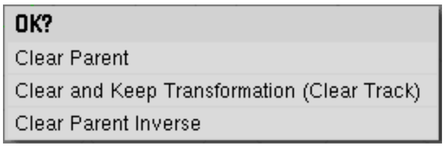
Move child

Move child

You can *Move* a child object to its parent by clearing its origin. The relationship between the parent and child still remains. Select the child object and press Alt O. By confirming the dialog the child object will snap to the parent's location. Use Outliner view to verify that the child object is still parented.

Remove relationship/Clear Parent

You can *Remove* a parent−child relationship via Alt P; see image (*Remove relationship*).



Remove relationship

The menu contains:

Clear Parent

> If the parent in the group is selected nothing is done. If a child or children are selected they are disassociated with the parent, or freed, and they return to their *original* location, rotation, and size.

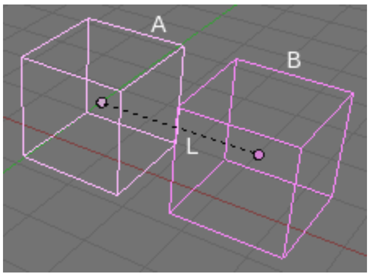
Clear and Keep Transformation (Clear Track)

> Frees the children from the parent, and *keeps* the location, rotation, and size given to them by the parent.

Clear Parent Inverse

> Places the children with respect to the parent as if they were placed in the Global reference. This effectively clears the parent's transformation from the children. For example, if the parent is moved 10 units along the X axis and "Clear Parent Inverse" is invoked, any selected children are freed and moved −10 units back along the X axis. The "Inverse" only uses the last transformation; if the parent moved twice, 10 units each time for a total of 20 units, then the "Inverse" will only move the child back 10 units not 20.

Parenting Examples

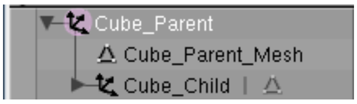


Parenting Example

The Active Object, in light pink (Cube A), will be made the parent of all the other object(s) in the group (darker pink/purple Cube B). The center(s) of all children object(s) are now linked to the center of the parent by a dashed line; see image (*Parenting Example*). The parent object is cube "**A**" and the child object is cube "**B**". The link is labelled "**L**".

At this point, grabbing, rotating, and scaling transformations to the parent will do the same to the children. Parenting is a very important tool with many advanced applications, as we'll see in later chapters; it is used extensively with advanced animations.

Hints



Outliner view

There is another way to see the parent−child relationship in groups and that is to use the Outliner view which is described in The Ouliner window. Image (*Outliner view*) is an example of what the Outliner view looks like for the (*Parenting Example*). Cube "**A**"'s object name is "Cube_Parent" and cube "**B**" is "Cube_Child".

Seperating Objects

Mode: Edit Mode

Hotkey: P

Menu: Mesh â†’ Vertices â†’ Seperate

Description

At some point, you'll come to a time when you need to cut parts away from a mesh to be seperate, but you might wonder how to do that. Well, the operation is easy.

General

To seperate an object, the vertices (or faces) must be selected and then seperated, though there are several different ways to do this.

Options

Selected

This option separates the selection to a new object.

All Loose Parts

Suzanne decapitated neatly

Separates the unselected part of the mesh.

By Material

Creates seperate mesh objects for each material.

Grouping objects

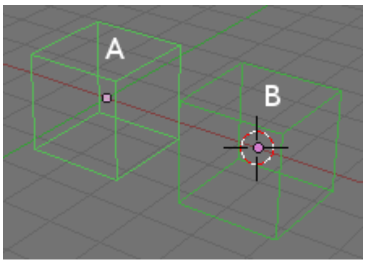
Mode: Object mode

Panel: Object â†’ Object and Links

Hotkey: Ctrl G

Menu: Object â†’ Parent â†’ Add to New Group

Description



Grouped objects

Group objects together without any kind of transformation relationship. Use groups to just logically organize your scene, or to facilitate one−step appending or linking between files or across scenes. Objects that are part of a group always shows as light green when selected; see image (*Grouped objects*).

Options

Adding to or Creating Group

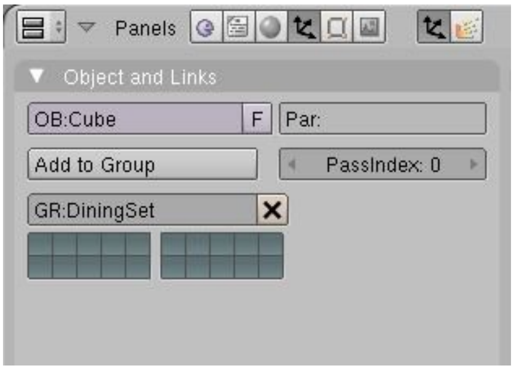
Groups pop−up

Ctrl G pops up a dialog for adding to existing groups or creating new a group; see image (*Groups pop−up*).

Alternatively, with the object selected or in Edit Mode, click the Add to Group button shown above in image (Naming a Group). The popup list allows you to click on an existing group, or create a new one.

You can also ostracize, or banish, the selected object from all groups by selecting the Remove option.

Naming a Group



Naming a Group

There are many ways to create a Group. The simplest is to shift−select all the objects you want to be grouped together, and then, in Object Mode, find ObjectGroupAdd to New Group in the menu of the 3D area or use the Ctrl G shortcut. The selected objects will now have a green outline indicating that they are part of one or more groups. Then, among the Buttons Window Object (F7) context, in the Object and Links panel, in the GR: field shift click and type DiningSet. save and close the file.

To name groups in the Outliner window, select Groups as the outliner display from the header combo box, and Ctrl LMB click on the group name. The name will change to an editable field; make your changes and press Enter.

Restricting Group Contents via Layers

That cluster of layer buttons below a Group designation determines from which layers the group objects will be included when duped. If your Group contains objects on layers 10, 11 and 12, but you disable the layer 12 button in the Group controls, duplicates of that group (using the Dupligroup feature) will only show the portions of the group that reside in layers 10 and 11.

Appending or Linking Groups

To append a group from another .blend file, consult this page. In summary, FileAppend or Link(filename)Group<groupname>.

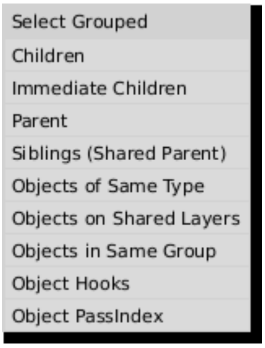
Select Grouped

Mode: Object mode

Hotkey: Shift G

Menu: Select â†’ Grouped

Description



Selected Grouped pop−up

Shift G pops up a dialog for selecting objects based on parenting and grouping characteristics; see image (*Selected Grouped pop−up*).

Options

Children

> Selects all the active object's children, and the children's children, up to the last generation.

Immediate Children

> Selects all the active object's children but not those of the selected object's parent.

Parent

> Selects the parent of the active object and deselects the active object.

Siblings (Shared Parent)

> Selects all the siblings of the Active Object.

Objects of Same Type

> Select objects based on the current object type.

Objects on Shared Layers

> This actually has nothing to do with parents. It selects all objects on the same layer(s) of the active object.

Objects in Same Group

Select objects that belong to the same group as the selected object(s).

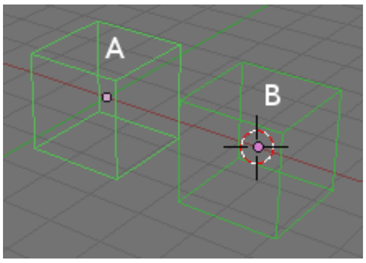
Object Hooks

Select all Hooks which are attached to the Active Object.

Object PassIndex

Select all objects which have the same PassIndex number as the Active Object. See ID Mask Node usage for more information on this option.

Examples



Grouped objects

(*Grouped objects*) shows to cubes grouped together where **A** is the last selected object indicated by being drawn in a lighter color.

Previous: Manual/Editing Objects	Contents	*Next:* Manual/Duplication

Duplicate

Mode: Edit mode / Object mode

Hotkey: Shift D

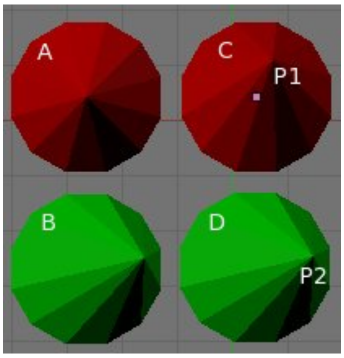
Menu: Object â†’ Duplicate

Description

This will create a visually identical copy of the selected object(s). The copy is created at the same position as the original object and you are automatically placed in Grab mode. Reference (*Duplicate Example*) for the discussions below.

This is a new object and it "**shares**" all the Material(s), Texture(s), and IPO(s) from the original object; this is sometimes called a shallow link because the object's mesh information and transform properties are separate copies.

Examples



Duplicate Example

The cone labeled "**C**" is a *Duplicate* of cone "**A**". Here are some properties to notice:

- The vertex at "**P1**" has been moved but the same vertex on cone "**A**" is unchanged. This means the mesh data are copies not links.
- Cone "**C**"'s color is red because cone "**A**"'s color is red. This means the material properties are linked not copied.
- If you rotate cone "**C**" cone "**A**" remains unchanged. This means the transform properties are copies not links.

If you want separate copies you need to manually create them, one for each corresponding property. You can make separate materials for each, as described in the Materials Chapter.

Linked Duplicates

Mode: Edit mode / Object mode

Hotkey: Alt D

Menu: Object â†’ Duplicate Linked

Description

You also have the choice of creating a *Linked Duplicate* rather than a *Duplicate*; this is called a deep link. This will create a new object with **all** of its data linked to the original object. If you modify one of the linked objects in EditMode, all linked copies are modified. Transform properties still remain copies not links so you still can rotate, scale, and move freely without affecting the other copy. Reference (*Duplicate Example*) for the discussions below.

Examples

The cone labeled "**D**" is a *Linked Duplicate* of cone "**B**" using Alt D. Here are some properties to notice:

- The vertex at "**P2**" has moved and the same vertex on cone "**B**" has moved as well. This means the mesh data are links not copies.
- Cone "**D**"'s color is green because cone "**B**"'s color is green. This means the material properties are also linked and not copied.
- If you rotate cone "**D**" cone "**B**" remains unchanged. This means the transform properties are copies not links.

A common table has a top and four legs. Model one leg, and then make linked duplicates three times for each of the remaining legs. If you later make a change to the mesh, all the legs will still match. Linked duplicates also apply to a set of drinking glasses, wheels on a car; anywhere there is repetition or symmetry.

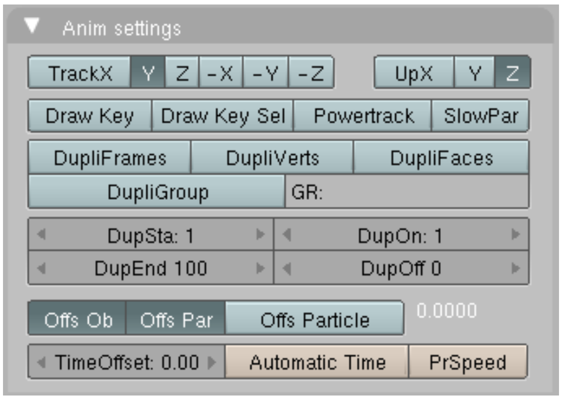
Procedural Duplication

Mode: Object mode / Edit Mode

Panel: Anim Settings

Hotkey: F7

There are currently four ways in Blender to procedurally duplicate objects. These options are located in the Objects context (F7) buttons, panel Anim Settings.



Anim settings panel

DupliVerts
This creates an instance of all children of this Object on each vertex (for Mesh−Objects only).
DupliFaces
This creates instances of all children of this Object on each face (for Mesh−Objects only).
DupliGroup
This creates an instance of the group with the transformation of the Object. Group duplicators can be animation using Actions, or can get a Proxy
to do

− link to proxy page of the manual.

DupliFrames
For animated Objects, this creates an instance on every frame.

Hints

If you want Transform properties to be linked see the section on Parent Grouping.

Previous: Manual/Groups and Parenting	Contents	*Next:* Manual/Tracking

User Manual: Contents | Guidelines | Blender Version **2.42**

Tracking

Mode: Object mode

Panel: Object â†’ Constraints

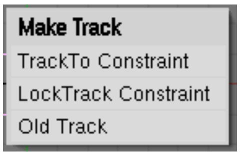
Hotkey: Ctrl T

Menu: Object â†’ Track â†’ Make Track

Description

Tracking consists of one object watching another. The watcher is the "Tracker" and the watched is the "Target". If the target moves the tracker rotates; if the tracker moves the tracker rotates. In both cases the tracker maintains a constant heading towards the target.

Options



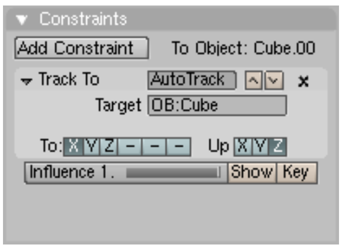
Make Track Menu.

To make one or more objects track another object (the target) select at least two objects and press CtrlT. The active object becomes the target and the others objects the trackers. The (*Make Track Menu*) provides several options for creating the initial tracking:

TrackTo Constraint

This options causes the tracker to point a local "To" axis at the target with an "Up" axis always maintaining a vertical orientation. This tracking is similar to billboard tracking in 3D. This is the preferred method over Old Track. As mentioned TrackTo constraint is the preferred tracking constraint because it has a more easily controlled constraining mechanism. It also can act as a constraint on the constraint stack and can be moved up and down in the stack.

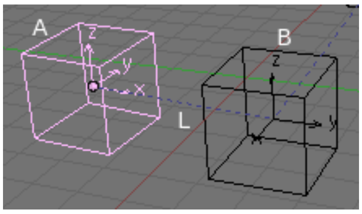
The controls for changing the "Tracking" and "Up" axis of a tracker object are located in the Constraints panel. This panel is located in the same place as the Anim settings and Draw panels (in Buttons Window, Object(F7) context, Object buttons sub−context); see (*Constraints panel*). The constraint fields are:



Constraints panel

- Target − The name of the target that the tracking object tracks.
- To − The tracking axis. It shouldn't be the same as the "Up" axis.
- Up − The "Up" oriented axis relative to the global coordinate system.
- Influence − This controls how accurately the tracking object tracks the target. "**0**" means that the constraint is turned off. The tracking object will remain locked in orientation. "**1**" means tracking is completely on and the tracking axis will stay tightly focused on the target.
- Show − This adds an influence IPO channel to the constraint if one is not present. You can then add keys to the channel.
- Key − This adds animation keys to the influence IPO channel. This is a very powerful combination. For example, you could have a camera with a TrackTo constraint applied and have input driving the influence channel.

If you select an invalid combination of "Tracking" and "Up" axis the field labeled "AutoTrack" will turn red. In addition, the tracking object will stop tracking the target until you choose a valid combination. This behavior is different than Old Track where Old Track would continue to track using a previous valid combination.



TrackTo example

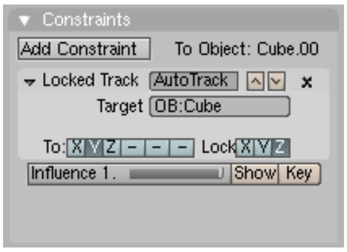
(*TrackTo example*) is an example of a cube using the TrackTo constraint. Cube "**A**" is tracking "**B**" where "**L**" is the tracking line. Notice how the tracking object's local axes are visible by using the Draw panel's axis button. You can clearly see the "Tracking" and "Up" axis. Cube "**A**"'s constraint setting are reflected in (*Constraints panel*). +X is the "Tracking" axis and +Z is the "Up" axis.

You can also see in (*Constraints panel*) what cube "**A**" is tracking by looking at the *Target* field. We can see that cube "**A**" is tracking cube "**B**" because cube "**B**"'s name is "OB:Cube". You can redirect tracking to another object simply by entering in the name of another object.

LockTrack Constraint

This options causes the tracker to point a local "To" axis at the target with a "Lock" axis always fixed and unable to rotate. This tracking is similar to billboard tracking in 2D. This constraint always has one axis locked such that it can not rotate. An example of billboarding is to have a Plane object textured with a tree image that always faces the camera.

The controls for changing the "Tracking" and "Lock" axis are located in the Constraints panel. This panel is located in the same place as the Anim settings and Draw panels; see (*Constraints panel*). The constraint fields are:



Locked Constraints panel

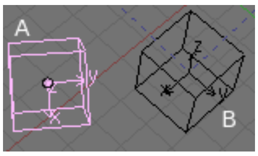
- Target − The name of the target that the tracking object tracks.
- To − The tracking axis. It shouldn't be the same as the "Lock" axis.
- Lock − The locked axis relative to the global coordinate system. The tracking object's local axis will snap to a global axis.
- Influence − This controls how accurately the tracking object tracks the target. "**0**" means that the constraint is turned off. The tracking object will remain locked in orientation. "**1**" means tracking is completely on.
- Show − This adds an influence IPO channel to the constraint if one is not present. You can then add keys to the channel.
- Key − This adds animation keys to the influence IPO channel. This constraint works well for billboarding 2D trees.

Note

According to the documentation the Lock axis buttons of the LockTrack Constraint command, when pressed will take the selected Lock axis and point it toward the global axis of the same type. This doesn't seem to happen in Blender 2.46 as the Lock axis does not appear to move to orient along the global axis.

For further details see TrackTo constraint. LockTrack and TrackTo are very similar where the former has a locked axis verses an "Up" axis.

Old Track



Old Track constraint

This is an older algorithm prior to version 2.30 and is similar to TrackTo Constraint in that no axis is locked. This algorithm merely tries to keep a "To" axis pointed at the target. The tracking object will usually end up in an odd orientation when this constraint is first applied. In order to get correct results use Alt R when applying or changing the tracking or "Up" axis. However, the preferred method to use is TrackTo Constraint. Let's assume you have selected Old Track in the dialog with two cubes selected; see (*Old Track constraint*). By default the inactive object(s) track the active object so that their local +Y axis points to the tracked object. Cube "**A**" is tracking cube "**B**" using the Old Track constraint. You can see that "**A**"'s +Y axis is pointing at "**B**" but at an odd orientation. This typically happens if the object already has a rotation of its own. You can produce correct tracking by canceling the rotation on the tracking object using (Alt R).

The orientation of the tracking object is also set such that the chosen "Up" axis is pointing upward.

Setting track axis.

If you want to change this you need to get to the "Anim settings" panel where Old Track's setting are accessed. First select the tracking object (not the target) and change the *Button Window* to *Object Context* by clicking the icon (), or F7; see (*Setting track axis*).

You then have the option of selecting the *Tracking axis* from the first column−set of six radio buttons and/or selecting the *upward−pointing* axis from the second column−set in the Anim Setting panel. Each time you change the "Up" axis you need to apply (Alt R) otherwise the tracking object will continue to track with the old orientation. This is one of the drawbacks to using Old Track.

To clear or remove a track constraint, select the tracking object and press Alt T. As with clearing a parent constraint, you must choose whether to lose or save the rotation imposed by the tracking. Note: (Alt R) only works with the Old Track constraint.

Hints

The *active* object always becomes the target object to be tracked. In all but Old Track a blue dashed line is drawn between the tracker and target indicating that a tracking constraint is in place between the corresponding objects. If you see an object tracking another object without a dashed blue line then you know the tracking object is using the Old Track constraint.

Invalid Tracking or settings

If you choose an invalid "Tracking axis" and/or "Up" axis the tracking object keeps it current orientation and ignores the incorrect selections. For example, if you choose the +Z axis as the tracking axis and also choose the +Z axis as the "Up" axis you have choosen an invalid combination because you can't have the tracking object's +Z axis doing two different things at the same time.

If you have problems setting up the correct "Tracking" and "Up" axes you may want to turn on the tracking object's local axes. You can do this from the Draw panel by clicking on the "Axis" button. see The Interface chapter for further details on the Draw panel.

Previous: Manual/Duplication	Contents	*Next:* Manual/Meshes

User Manual: Contents | Guidelines | Blender Version **2.4x**

Edit Mode

You can work with geometric objects in two modes: Object Mode and Edit Mode. Operations in Object Mode affect whole objects, and operations in Edit Mode affect only the geometry of an object, but not its global properties such as location or rotation. You switch between these two modes with the TAB key.

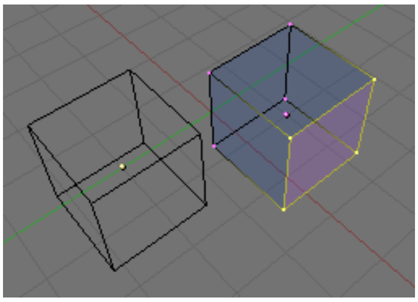
Object Mode is recognizable if you see the following header in the 3D view:

Object Mode header.

Edit Mode is recognizable if you see the following header in the 3D view:

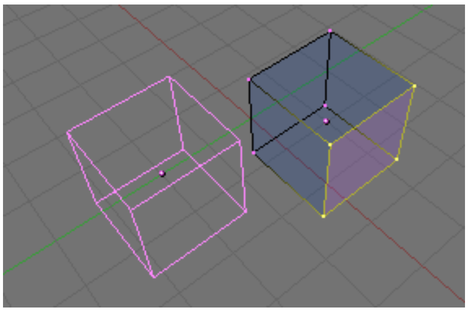
Edit Mode header.

After creating an object you are immediately placed in Edit Mode. Edit Mode only works on one object at a time, the Active Object. An object outside Edit Mode (i.e. Object Mode) is drawn in purple in the 3D Viewport Windows (in wireframe mode) when selected; it is black otherwise.



One cube selected.

In Edit Mode each vertex is drawn in purple, each edge is drawn in black and each face is drawn in translucent dark−blue. In image (*One cube selected*) the Cube on the right is in Edit Mode. The Cube on the left is in Object Mode *and* not selected. Each selected vertex or edge is highlighted in yellow.

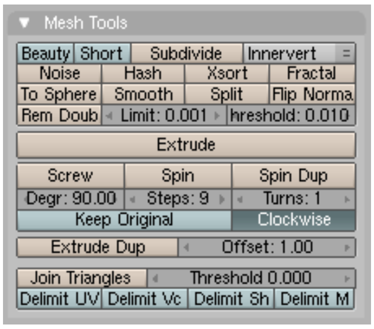


Two Cubes selected prior to Edit Mode.

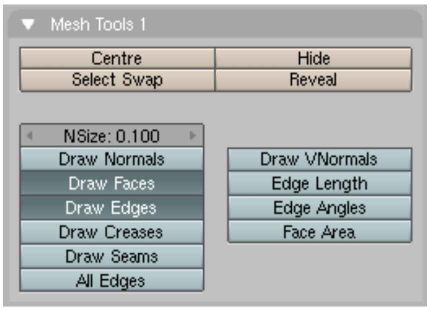
If multiple objects are selected and Edit Mode is entered then the last object selected (the Active Object) enters Edit Mode. The other objects remain purple and in Object Mode. As shown in image (*Two Cubes selected prior to Edit Mode*), both cubes were selected prior to Edit Mode and now the left cube is still purple and the right cube (the Active Object) is in Edit Mode.

If enough vertices are selected to form a face then that face is highlighted in translucent purple while the remaining faces are highlighted in translucent dark−blue. This helps give you a frame of reference when selecting vertices, edges or faces. The translucent effect indicates that you have selected enough vertices to imply one or more faces. See [Edge and Face Tools] for further details on implicit selections.

If the Buttons Window is visible and Editing Context button (F9) is activated then two panels appear while in Edit Mode (see images *Mesh Tools* and *Mesh Tools 1*):



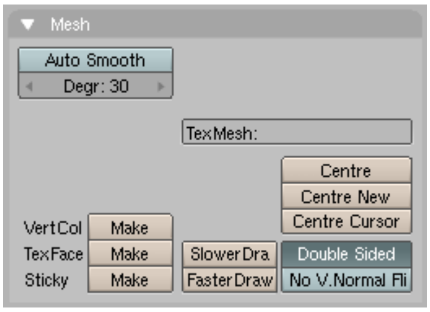
Mesh Tools



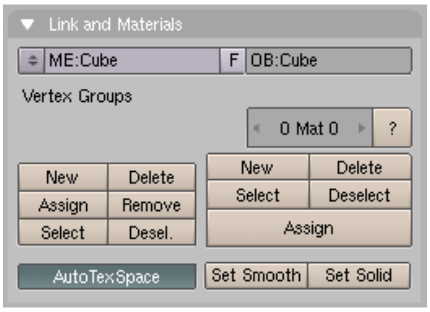
Mesh Tools 1

By default the buttons (Draw Faces and Draw Edges) are pre−selected and any selected edges and faces are highlighted.

In addition, panels (see images *Link and Materials* and *Mesh Tools*) are updated.



Mesh Tools

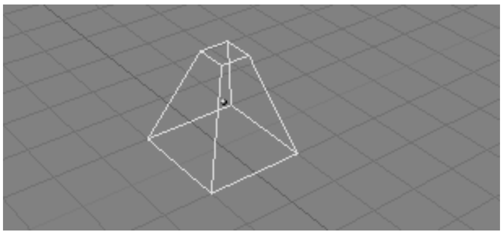


Link and Materials

The Link and Materials panel gains the New, Delete, Assign, Remove, Select and Desel. buttons. The Mesh panel loses the Decimator, Apply and Cancel group of buttons.

Basic Editing

Most simple operations from Object Mode (like selecting, moving, rotating, and scaling) work the same way on vertices as they do on objects. Thus, you can learn how to handle basic Edit Mode operations very quickly. The only notable difference is a new scaling option, Alt S which scales the selected vertices along the direction of the Normals (shrinks−fattens). The truncated pyramid in image (*Chopped−off pyramid*), for example, was created with the following steps:



Chopped−off pyramid

- Add a cube to an empty scene. If not in Edit Mode then use TAB to enter Edit Mode.
- Make sure all vertices are deselected (purple). Use border select (B) to select the upper four vertices.
- Check that the scaling center is set to *anything but* the 3D Cursor (you *don't* want to see as the selected pivot point), then switch to scale mode (S), reduce the size, and confirm with LMB.
- Exit Edit Mode by pressing TAB.

All operations in Edit Mode are ultimately performed on the vertices; the connected edges and faces automatically adapt, as they depend on the vertice's positions. To select an edge, you must select the two endpoints or place the mouse on the edge and press Alt RMB . To select a face, each corner must be selected.

Edit Mode operations are many, and most are summarized in the Editing Context Buttons window, accessed via the ( ) header button or via F9 (*Edit Context*).

Mirror Axis and Modifier

Mirror Axis

One extra feature for Edit Mode is the Mirroring tool. If you have some vertices selected and you press M you will be presented with a Menu containing nine options (*Mirror Axis*). You can select from these to mirror the selected vertices with respect to any of the X, Y or Z axes of the Global, Local, or Viewing reference. If you need to select groups of vertices use the handy Circle Select tool.

Editor's Note

There is a much more advanced tool for performing mirroring operations and that is the Mirror Modifier

Specials

With W you can call up the *Specials* menu in Edit Mode, see image (*Specials Menu*). With this menu you can quickly access functions which are frequently required for polygon−modelling.

Specials Menu.

- Subdivide − Each selected edge is split in two, new vertices are created at middle points, and faces are split too, if necessary.
- Subdivide Multi − This is identical to Subdivide except a dialog pops up asking for the *number of cuts* or repeated sub−divisioning. The default is "**2**".
- Subdivide Multi Fractal − As above, but new vertices are randomly displaced within a user−defined range.
- Subdivide Smooth − Same as Subdivide, but new vertices are displaced towards the barycenter (centre of mass) of the connected vertices.
- Merge − Merges selected vertices into a single one, at the barycenter position or at the 3D Cursor position.
- Remove Doubles − Merges all of the selected vertices whose relative distance is below a given threshold (0.001 by default).
- Hide − Hides selected vertices.
- Reveal − Shows hidden vertices.
- Select Swap − All selected vertices become unselected and vice−versa.
- Flip Normals − Change the Normal directions of the selected faces.
- Smooth − Smooths out a mesh by moving each vertex towards the barycenter of the linked vertices.
- Bevel − Bevels the entire object regardless of the selected vertices, edges or faces. See Manual/Edge_and_Face_Tools#Bevel
- Set Smooth − Changes the selected faces to smoothing shading.
- Set Solid − Changes the selected faces to faceted or flat shading.

Keyboard Tip:

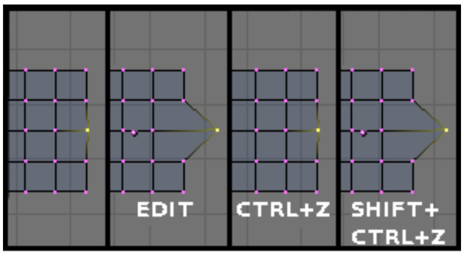
You can access the entries in a PopupMenu by using the corresponding numberkey. For example, pressing W and then 1 will subdivide the selected edges without you having to touch the mouse at all.

Mesh Tools Panel

Many of these actions have a button of their own in the Mesh Tools panel of the Edit Buttons Window (*Edit Context*) see image (Mesh Tools Panel). The Remove doubles threshold can be adjusted on that panel too.

Mesh Undo

Blender has a global undo system, giving full multiple level undo capabilities in all areas of Blender. Exceptions are: Edit mode Armature, and Fileselect, Audio and Oops windows.

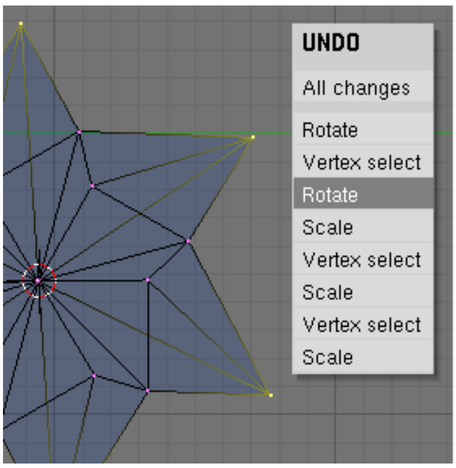


Undo and Redo

The new global hotkey for undo is Ctrl Z, and Shift Ctrl Z for redo.

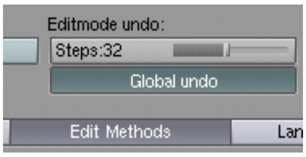
Mesh undo works in the background saving copies of your mesh in memory as you make changes. Pressing the Ctrl Z in mesh Edit Mode reverts to the previously saved mesh, undoing the last edit operation (*Undo and Redo*).

Mesh Undo operations are only stored for one mesh at a time. You can leave and re−enter Edit Mode for the same mesh without losing any undo information, but once another mesh is edited, the undo information for the first is lost. Pressing Shift Ctrl Z re−does the last undo operation (*Undo and Redo*).



Undo Menu

Pressing Alt U brings up the Undo menu see image (*Undo Menu*). This lists all the undo steps by name so you can quickly find your way back to a known good point in your work. The *Undo Menu* also contains the option All Changes. This option is more powerful than merely pressing Ctrl Z repeatedly, and will reload the mesh data as it was at the beginning of your edit session, even if you have used up all your undo steps.



User Preferences/Edit Methods

Mesh Undo has the potential to be very memory intensive. A mesh of 64,000 faces and vertices can use over 3MBs of RAM per undo step! If you are on a machine that is strapped for RAM (Memory), in the User reference Window under Edit Methods, there is a NumButton see image (**Editmode undo**) for setting the maximum number of undo steps saved see image (*User Preferences/Edit Methods*). The allowable range is between 1 and 64. The default is 32.

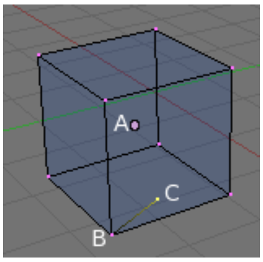
Previous: Manual/Tracking	Contents	*Next:* Manual/Mesh Structures

User Manual: Contents | Guidelines | Blender Version **2.4x**

Mesh Structures: Vertices, Edges and Faces

In basic meshes, everything is built from three basic structures: *Vertices*, *Edges* and *Faces*. (We're not talking about Curves, NURBS, and so forth here.) But there is no need to be disappointed: This simplicity still provides us with a wealth of possibilities that will be the foundation for all our models.

Vertices



Vertex example.

A vertex is primarily a single point or position in 3D space. It is usually invisible in rendering and in ObjectMode. Don't mistake the center point of an object for a vertex. It looks similar, but it's bigger and you can't select it. (*Vertex example*) shows the center point labeled as "**A**". "**B**" and "**C**" are vertices.

To create a new vertex, change to Edit mode, hold down CTRL, and click with the LMB. Of course, as a computer screen is two−dimensional, Blender can't determine all three vertex coordinates from one mouse click, so the new vertex is placed at the depth of the 3D cursor 'into' the screen. Any vertices selected previously are automatically connected to the new one with an edge. Vertex labeled "**C**" is a new vertex added to the cube with a new edge (**B** to **C**)

Edges

An edge always connects two vertices with a straight line. The edges are the 'wires' you see when you look at a mesh in wireframe view. They are usually invisible on the rendered image. They are used to construct faces. Create an edge by selecting two vertices and pressing F.

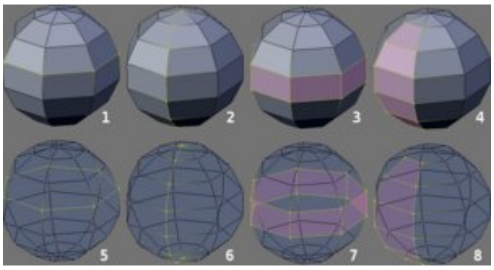
Faces

A Face is the highest level structure in a mesh. Faces are used to build the actual surface of the object. They are what you see when you render the mesh. A Face is defined as the area between either three (triangles) or four vertices (quads), with an Edge on every side. Triangles always work well, because they are always flat and easy to calculate.

Take care when using four−sided faces (quads), because internally they are simply divided into two triangles each. Four−sided faces only work well if the Face is pretty much flat (all points lie within one imaginary plane) and convex (the angle at no corner is greater than or equal to 180 degrees). This is the case with the faces of a cube, for example. That's why you can't see any diagonals in its wireframe model, because they would divide each square face into two triangles.

While you could build a cube with triangular faces, it would just look more confusing in Edit mode. An area between three or four vertices, outlined by Edges, doesn't have to be a face. If this area does not contain a face, it will simply be transparent or non−existent in the rendered image. To create a face, select three or four suitable vertices and press F.

Mesh Structures: Edge Loops and Face Loops



Edge and Face Loops

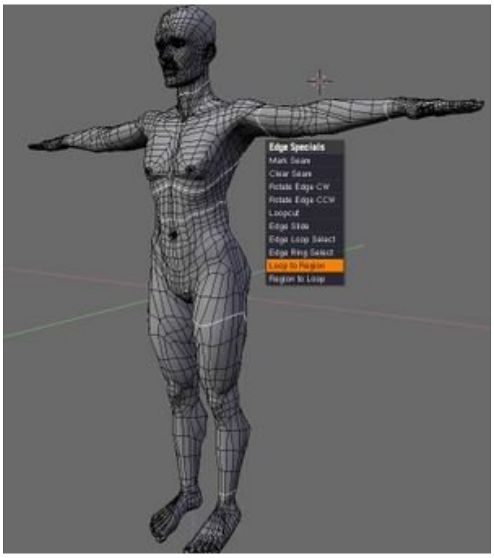
Edge and Face Loops are sets of Faces or Edges that form continuous "loops" as shown in (*Edge and Face Loops*). The top row (1−4) shows a solid view, the bottom row (5−8) a wireframe view of the same loops.

Note that loops 2 and 4 do not go around the whole model. Loops stop at so called poles because there is no unique way to continue a loop from a pole. Poles are vertices that are connected to either three or five or more edges. Accordingly, vertices connected to exactly one, two or four edges are not poles. Loops that do not end in poles are cyclic (1 and 3). They start and end at the same vertex and divide the model into two partitions.

Loops can be a quick and powerful tool to work with specific, continuous regions of a mesh and are a prerequisite for organic character animation. For a detailed description of how to work with loops in Blender please refer to the Manual page on Edge and Face Tools.

Edge Loops

Loops 1 and 2 in (*Edge and Face Loops*) are Edge Loops. They connect vertices so that each one on the loop has exactly two neighbors that are not on the loop and placed on both sides of the loop (except the start and end vertex in case of poles).



Edge Loops in organic modeling

Edge Loops are an important concept especially in organic (subsurface) modeling and character animation. When used correctly, they allow you to build models with relatively few vertices that look very natural when used as subdivision surfaces and deform very well in animation.

Take (*Edge Loops in organic modeling*) as an example: The Edge loops follow the natural contours and deformation lines of the skin and the underlying muscles and are more dense in areas that deform more when the character moves, for example at the shoulders or knees.

Further details on working with Edge Loops can be found in [Edge Loop Selection].

Face Loops

These are a logical extension of Face Loops in that they consist of the faces between two Edge Loops, as shown in loops 3 and 4 in (*Edge and Face Loops*). Note that for non−circular loops (4) the faces containing the poles are not included in a Face Loop.

Further details on working with Face Loops can be found in [Face Loop Selection].

Previous: Manual/Meshes	Contents	*Next:* Manual/Mesh Primitives

User Manual: Contents | Guidelines | Blender Version **2.44**

Basic Mesh Objects

Mode: Object Mode

Hotkey: Shift A

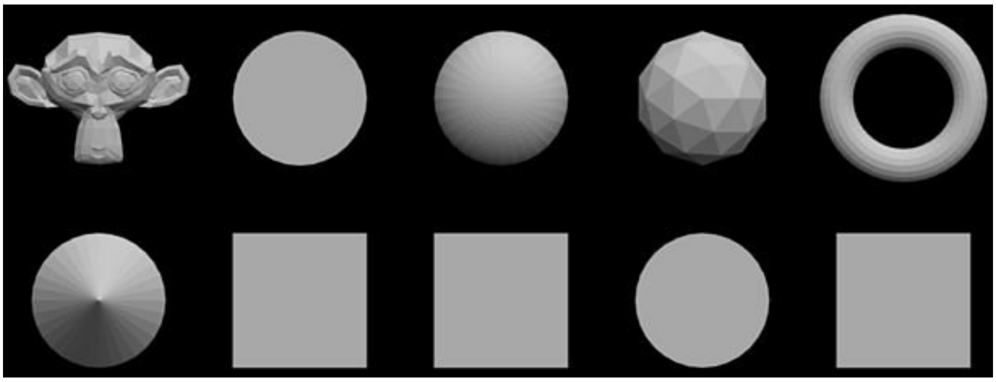
Menu: Add Mesh (see Add Submenu)

Description

A common object type used in a 3D scene is a Mesh. Blender comes with a number of 'primitive' mesh shapes that you can start modelling from. There are several options that are included in more than one primitive that you should know. The first is Radius. This option is included with the Circle, Cylinder, Cone, UVSphere and IcoSphere primitives and sets their starting size. And the 2nd option is Depth, a parameter that comes with the Cylinder and Cone that sets their starting size.

Options

Basic Objects shows the variety of basic Mesh objects that can be created.



All the mesh primitives (none smoothed)

Plane

A standard plane contains four vertices, four edges, and one face. It is like a piece of paper lying on a table; it is not a real three−dimensional object because it is flat and has no thickness. Objects that can be created with planes include floors, tabletops, or mirrors.

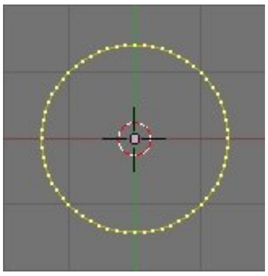
Note

You can make the mesh three−dimensional by moving one of more of the vertices out of the plane of the plane.

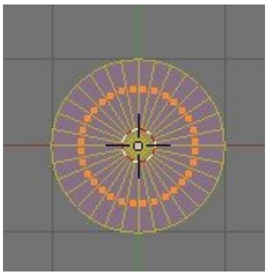
Cube

A standard cube contains eight vertices, 12 edges, and six faces, and is a real three−dimensional object. Objects that can be created out of cubes include dice, boxes, or crates.

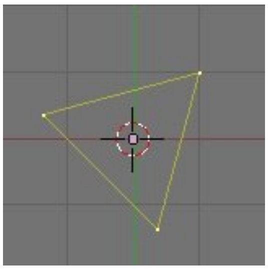
Circle



A circle with 64 vertices gives a smooth circle.



Filled circle with 32 vertices.



A circle with only 3 vertices is actually a triangle.

A standard circle is comprised of *n* vertices. The number of vertices and radius can be specified in the popup window which appears when the circle is created, shown in 'Add circle popup window'. When 'Fill' button is active, the circle will be filled with triangular faces which share a vertex in the middle. However the circle is only a flat shape. If it is not filled and you want to render it, you must assign it a wireframe material (Shading (F5) context, Material buttons sub−context, Links and Pipeline panel and finally the Wire button). The 'Radius' parameter adjusts the size of the circle.

Add circle popup window

The more vertices the circle contains, the smoother its contour will be (see 'Circle' and 'Triangle').

Note

You can make the mesh three−dimensional by moving one or more of the vertices out of the plane of the circle.

UVSphere

A standard UVsphere is made out of *n* segments and *m* rings. The level of detail and radius can be specified in the popup window which appears when the UVsphere is created. Increasing the number of segments and rings makes the surface of the UVsphere smoother. Segments are like Earth's meridians, going pole to pole and rings are like Earth's parallels. Example objects that can be created out of UVspheres are balls, heads or pearls for a necklace.

Add UVSphere popup window

Note

If you specify a six segment, six ring UVsphere you'll get something which, in top view, is a hexagon (six segments), with five rings plus two points at the poles. Thus, one ring fewer than expected, or two more, if you count the poles as rings of radius 0.

Icosphere

An Icosphere is made up of triangles. The number of subdivisions and radius can be specified in the window that pops up when the Icosphere is created; increasing the number of subdivisions makes the surface of the Icosphere smoother. At level 1 the Icosphere is an icosahedron, a solid with 20 equilateral triangular faces. Any increasing level of subdivision splits each triangular face into four triangles, resulting in a more spherical appearance. Icospheres are normally used to achieve a more isotropical and economical layout of vertices than a UVsphere.

Add icosphere popup window

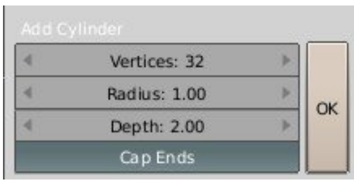
Note

It is possible to add an icosphere subdivided 500 times. Adding such a dense mesh is a sure way to cause a program crash. An icosphere subdivided 10 times would have 5,242,880 triangles, so be very careful about this!

Cylinder

A standard cylinder is made out of *n* vertices. The number of vertices in the circular cross−section can be specified in the popup window that appears when the object is created; the higher the number of vertices, the smoother the circular cross−section becomes. The radius and depht parameters controls dimensions of cylinder. Objects that can be created out of cylinders include handles or rods.

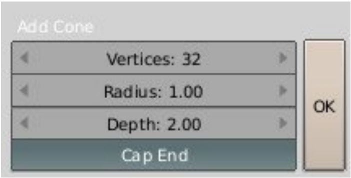
If 'Cap Ends' is inactive, the created object will be a tube. Objects that can be created out of tubes include pipes or drinking glasses. (The basic difference between a cylinder and a tube is that the former has closed ends.)



Add cylinder popup window

Cone

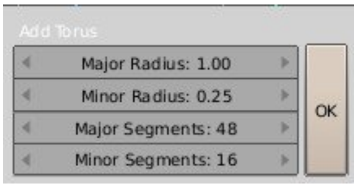
A standard cone is made out of *n* vertices. The number of vertices in the circular base, dimensions and option to close the base of cone can be specified in the popup window that appears when the object is created; the higher the number of vertices, the smoother the circular base becomes. Objects that can be created out of cones include spikes or pointed hats.



Add cone popup window

Torus

A doughnut−shaped primitive created by rotating a circle around an axis. The overal dimensions are defined by the major and minor radius. The number of vertices (in segments) can be different for the circles and is specified in the popup window with both radii (Major Segments and Minor Segments).



Add torus popup window

Grid

A standard grid is made out of *n* by *m* vertices. The resolution of the x−axis and y−axis can be specified in the popup window which appears when the object is created; the higher the resolution, the more vertices are created. Example objects that can be created out of grids include landscapes (with the proportional editing tool) and other organic surfaces. You can also obtain a grid when you create a plane then use a sudivide modifier in Edit mode.

Monkey

This is a gift from old NaN to the community and is seen as a programmer's joke or "Easter Egg". It creates a monkey's head once you press the Monkey button. The Monkey's name is *Suzanne* and is Blender's mascot. Suzanne is very useful as a standard test mesh much like the Utah Tea Pot or the Stanford Bunny.

Previous: Manual/Mesh Structures	Contents	*Next:* Manual/Mesh Smoothing

User Manual: Contents | Guidelines | Blender Version **2.4x**

Mesh smoothing

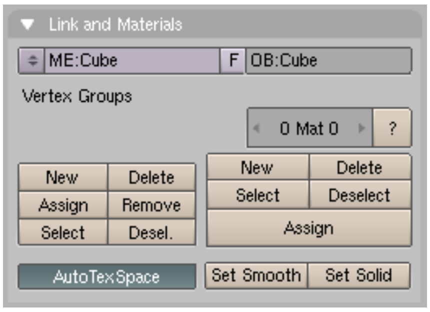
Simple un−smoothed test object

As seen in the previous sections, polygons are central to Blender. Most objects are represented by polygons and truly curved objects are often approximated by polygon meshes. When rendering images, you may notice that these polygons appear as a series of small, flat faces. See (*Simple un−smoothed test object*).

Sometimes this is a desirable effect, but usually we want our objects to look nice and smooth. This section shows you how to visually smooth an object, and how to apply the AutoSmooth filter to quickly and easily combine smooth and faceted polygons in the same object.

The last section shows the possibilities to smooth a mesh's geometry, not only its appearance.

Smoothing the entire mesh



Link and Materials

The easiest way is to set an entire object as smooth or faceted by selecting a mesh object, in ObjectMode, switching to the Editing Context (F9), and clicking the Set Smooth button in the Link and Materials panel shown in (*Link and Materials*).

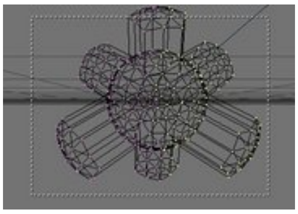
The button does not stay pressed, it forces the assignment of the "smoothing" attribute to each face in the mesh, also when you add or delete geometry. Now, rendering the image with (F12) should produce the image shown in (*Completely smoothed*).

Completely smoothed

Notice that the outline of the object is still strongly faceted. Activating the smoothing features doesn't actually modify the object's geometry; it changes the way the shading is calculated across the surfaces, giving the illusion of a smooth surface. Click the Set Solid button in the same panel to revert the shading back to that shown in (*Simple un−smoothed test object*) above.

Smoothing parts of a mesh

Manually

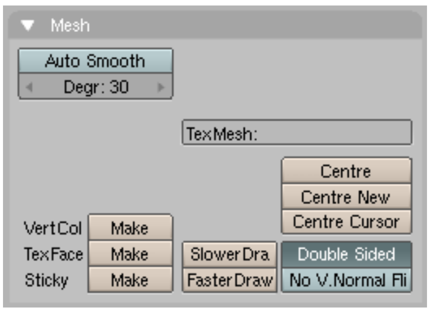


Object in editmode with some faces selected.

Alternatively, you can choose which faces to smooth by entering Edit Mode for the object with TAB, then selecting the faces and clicking the Set Smooth button (*Object in editmode with some faces selected.*). The selected faces are in marked in Yellow.

When the mesh is in *Editmode*, only the selected faces will receive the "smoothing" attribute. You can set solid faces (removing the "smoothing" attribute) in the same way by selecting faces and clicking the Set Solid button.

Autosmooth



AutoSmooth button group in the EditButtons window.

It can be difficult to create certain combinations of smooth and solid faces using the above techniques alone. Though there are work−arounds (such as splitting off sets of faces by selecting them and pressing Y), there is an easier way to combine smooth and solid faces, by using AutoSmooth.

Press the AutoSmooth button in the Mesh panel of the Edit Buttons (*AutoSmooth button group in the EditButtons window.*) to indicate which faces should be smoothed on the basis of the angle between faces (*Same test object with AutoSmooth enabled*). Angles on the model that are sharper than the angle specified in the Degr NumButton will not be smoothed. Higher values will produce smoother faces, while the lowest setting will look identical to a mesh that has been set completely solid.

Same test object with AutoSmooth enabled

Only faces that have been set as *smooth* will be affected by the AutoSmooth feature. A mesh, or any faces that have been set as solid will not change their shading when AutoSmooth is activated. This allows you extra control over which faces will be smoothed and which ones won't by overriding the decisions made by the AutoSmooth algorithm.

Smoothing the mesh geometry

The above techniques do not alter the mesh itself, only the way it is displayed and rendered. Instead of just making the mesh look like a smooth surface, you can also physically smooth the geometry of the mesh with these tools:

You can apply one of the following in edit mode:

- Smooth
- Subdivide Smooth
- Bevel

Alternatively, you can smooth the mesh non−destructively with one of the following modifiers:

- Smooth
  works like the Smooth tool in edit mode; can be applied to specific parts of the mesh using vertex groups
- Subdivision Surface
  Catmull−Clark subdivision produces smooth results. Sharp edges can be defined with subdivision creases or by setting certain edges to "sharp" and adding an EdgeSplit modifier (set to From Marked As Sharp) before the Subsurf modifier.

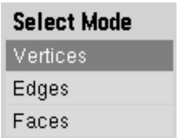
Previous: Manual/Mesh Primitives Contents *Next:* Manual/Selecting Meshes

User Manual: Contents | Guidelines | Blender Version **2.4**

There are many ways to select elements, and it depends on what mode you are in as to what selection tools are available. First we will go through these modes and after that a look is taken at basic selection tools.

Vertex, Edge and Face Modes

In EditMode there are three different selection modes.

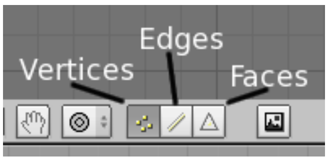


EditMode selection menu. Ctrl Tab

See (*EditMode selection menu*) for a picture of the popup menu.

- Vertices mode. Press Ctrl Tab and select Vertices from the popup menu. The selected vertices are drawn in yellow and unselected vertices are drawn in a pink color.
- Edges mode. Press Ctrl Tab and select Edges from the popup menu. In this mode the vertices are not drawn. Instead the selected edges are drawn in yellow and unselected edges are drawn in a black color.
- Faces mode. Press Ctrl Tab and select Faces from the popup menu. In this mode the faces are drawn with a selection point in the middle which is used for selecting a face. Selected faces are drawn in yellow with the selection point in orange, unselected faces are drawn in black.

Almost all modification tools are availiable in all three modes. So you can Rotate, Scale and Extrude etc. in all modes. Of course rotating and scaling a *single* vertex will not do anything useful, so some tools are more or less applicable in some modes.



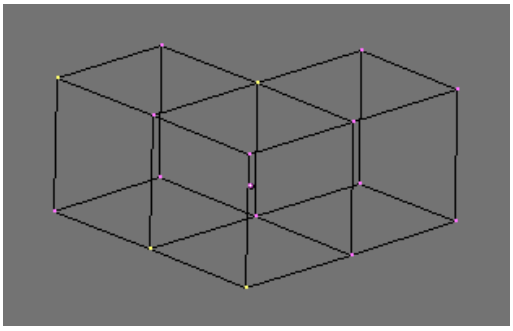
EditMode selection buttons.

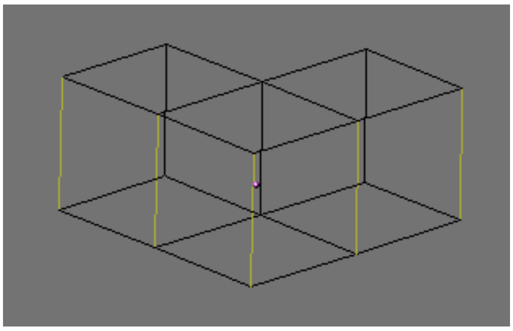
You can also enter the different modes by selecting one of the three buttons in the toolbar; see (*EditMode selection buttons*). Using the buttons you can also enter mixed modes by Shift LMB clicking the buttons.

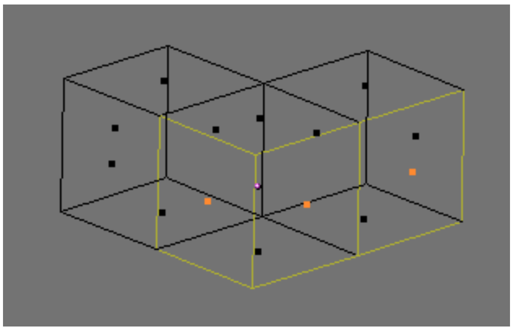
Note

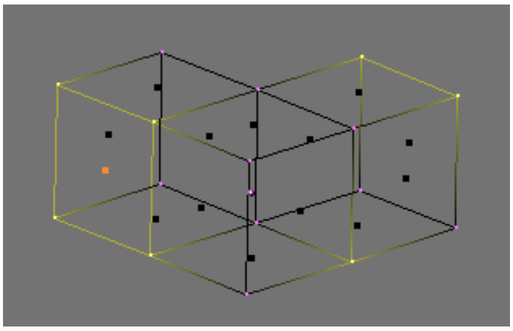
The Mode Selection buttons are only visible in EditMode.

When switching modes, from Vertices to Edges and from Edges to Faces, the selected parts will still be selected if they form a complete set in the new mode. For example, if all four edges in a face are selected, switching from Edges mode to Faces mode will keep the face selected. All selected parts that do not form a complete set in the new mode will be unselected. See *Vertex mode example.*, *Edge mode example.*, *Face mode example.* and *Mixed mode example.* for examples of the different modes.


Vertex mode example.


Edge mode example.


Face mode example.


Mixed mode example.

Point Selection

The most common way to select an element is to RMB on that item, this will replace the existing selection with the new item.

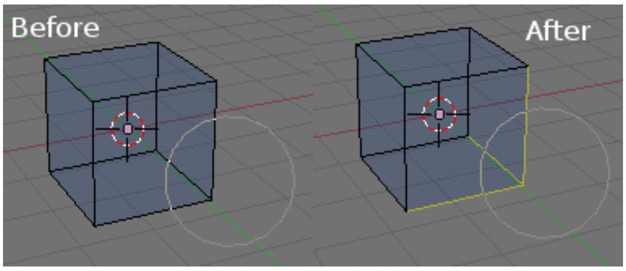
To add to the existing selection hold down Shift while right clicking. Clicking again on a selected item will de−select it.

Region selection allows you to select groups of elements within a 2D region. The region can be either a circle or rectangle. The circular region is only available in Edit mode. The rectangular region, or *Border Select*, is available in both Edit mode and Object mode.

Circular region

This selection tool is only available in Edit mode and can be activated with B, B. That is, pressing the B key twice in a row. Once in this mode the cursor changes to a dashed cross−hair with a 2D circle surrounding it. The tool will operate on whatever the current Select mode is. Clicking or dragging with the LMB , when elements are inside the circle, will cause those elements to be selected.

You can enlarge or shrink the circle region using NumPad + and NumPad − or the MW .



Circle Region Select.

(*Circle Region Select*) is an example of selecting edges while in Edge select mode. As soon as an edge intersects the circle the edge becomes selected. The tool is interactive such that edges are selected while the circle region is being dragged with the LMB.

If you want to de−select elements either hold MMB or Alt LMB and begin clicking or dragging again.

For *faces* the circle must intersect the face indicators usually represented by small pixel squares; one for each face.

To exit from this tool click RMB, or hit the Esc key.

Edit Mode

Face Select Mode

In Face select mode, faces can be selected based on whether they are *triangles*, *quads*, or *other*. Hotkeys:

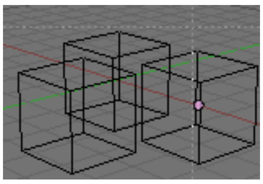
- Shift Ctrl Alt 3 selects all triangles
- Shift Ctrl Alt 4 selects all quads
- Shift Ctrl Alt 5 selects all non triangle/quad faces.

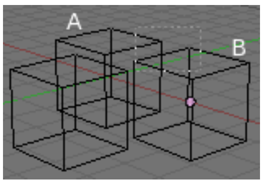
These tools are also available in both the 3D view header and Toolbox Select menus.

Rectangular or Border Select

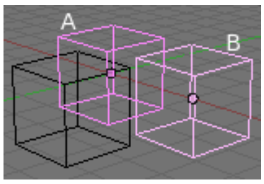
Border Select is available in either Edit mode or Object mode. To activate the tool use the B. Use Border Select to select a group of objects by drawing a rectangle while holding down LMB. In doing this you will select all objects that lie within or touch this rectangle. If any object that was last active appears in the group it will become selected *and* active.

In (*Start*) Border Select has been activated and is indicated by showing a dotted cross−hair cursor. In (*Selecting*), the *selection region* is being chosen by drawing a rectangle with the LMB. The rectangle is only covering cubes "**A**" and "**B**". Finally, by releasing LMB the selection is complete; see (*Complete*).


Start.



Selecting.



Complete.

Notice in (*Complete*) that cube "**B**" is also selected *and* active. This means that cube "**B**" was the last active object prior to using the Border Select tool.

Note

Border select adds to the previous selection, so in order to select only the contents of the rectangle, deselect all with A first. In addition, you can use MMB while you draw the border to deselect all objects within the

rectangle.

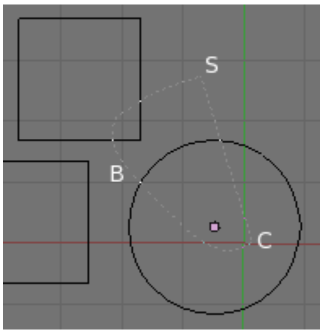
Lasso region

Lasso select is similar to Border select in that you select objects based on a region, except Lasso is a hand−drawn region that generally forms a circular/rounded shaped form; kind of like a lasso.

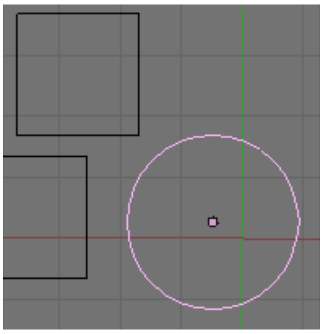
Lasso is available in either Edit mode or Object mode. To activate the tool use the Ctrl LMB while dragging. The one difference between Lasso and Border select is that in Object mode Lasso only selects objects where the lasso region intersects the objects center.

To de−select use Shift Ctrl LMB while dragging.

(*Selecting*) is an example of using the Lasso select tool. Dragging started at "**S**", curved around to "**B**" and stopped at "**C**". Notice that the lasso region included the circle's purple coloured object center.



Selecting.


Selected.

(*Selected*) is the result with the just the circle selected even though the square was in the lasso region.

Previous: Manual/Mesh Smoothing | Contents | *Next:* Manual/Basic Mesh Tools

User Manual: Contents | Guidelines | Blender Version **2.40**

Basic Mesh Modelling

In this section we will describe some of the most common Mesh editing tools: Extrude, Spin, Spin Dup, Screw, Warp and To Sphere.

Each tool is described using a simple tutorial. Extrude is explained by going through a simple set of steps for making a sword. Spin is explained by making a simple wine glass. Spin Dup is explained by making the hour mark on a clock face. Screw is explained by literally making a screw. And finally Warp is explained by warping some 3D text.

Extrude

Mode: Edit Mode Editing context F9

Panel: Mesh Tools Extrude

Hotkey: E

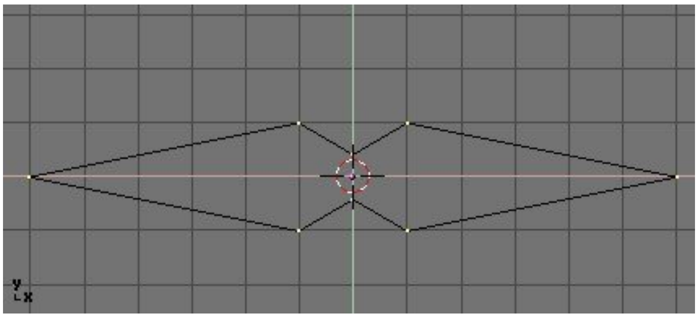
One tool of paramount importance for working with Meshes is the Extrude command (E). This command allows you to create cubes from rectangles and cylinders from circles, as well as easily create such things as tree limbs. Although the process is quite intuitive, the principles behind Extrude are fairly elaborate as discussed below.

- First, the algorithm determines the outside edge−loop of the Extrude; that is, which among the selected edges will be changed into faces. By default, the algorithm considers edges belonging to two or more selected faces as internal, and hence not part of the loop.
- The edges in the edge−loop are then changed into faces.
- If the edges in the edge−loop belong to only one face in the complete mesh, then all of the selected faces are duplicated and linked to the newly created faces. For example, rectangles will result in cubes during this stage.
- In other cases, the selected faces are linked to the newly created faces but not duplicated. This prevents undesired faces from being retained 'inside' the resulting mesh. This distinction is extremely important since it ensures the construction of consistently coherent, closed volumes at all times when using Extrude.
- Edges not belonging to selected faces, which form an 'open' edge−loop, are duplicated and a new face is created between the new edge and the original one.
- Single selected vertices which do not belong to selected edges are duplicated and a new edge is created between the two.

Grab mode is automatically started when the Extrude algorithm terminates, so newly created faces, edges, and vertices can be moved around with the mouse. Extrude is one of the most frequently used modelling tools in Blender. It's simple, straightforward, and easy to use, yet very powerful. The following short lesson describes how to build a sword using Extrude.

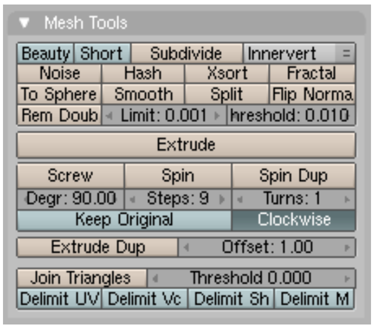
The Blade

- Start Blender and delete the default cube. In top view (NumPad 7) add a mesh circle with eight vertices. Move (G) the vertices so they match the configuration shown in *Deformed circle, to become the blade cross section.*.



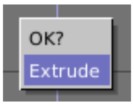
Deformed circle, to become the blade cross section.

- Select all the vertices (A) and scale them down with the S so the shape fits in two grid units. Switch to front view with NumPad 1.



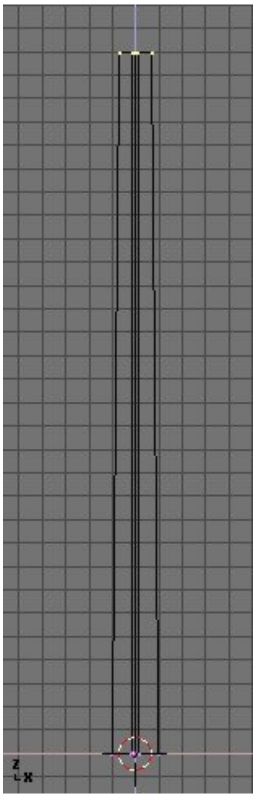
Extrude button in EditButtons context.

- The shape we've created is the base of the blade. Using Extrude we'll create the blade in a few simple steps. With all vertices selected press E, or click the Extrude button in the [Mesh Tools] Panel of the Editing Context (F9 − *Extrude button in EditButtons context.*).



Extrude confirmation box.

- A box will pop up asking Ok? Extrude (*Extrude confirmation box.*). Click this text or press Enter to confirm, otherwise move the mouse outside or press Esc to exit. If you now move the mouse you'll see that Blender has duplicated the vertices, connected them to the original ones with edges and faces, and has entered grab mode.

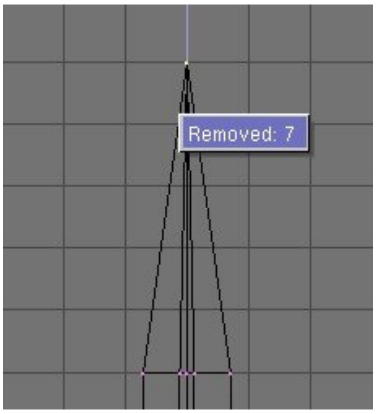


The Blade

- Move the new vertices up 30 units, constraining the movement with Ctrl, then click LMB to confirm their new position and scale them down a little bit with the S (*The Blade*).

- Press E again to extrude the tip of the blade, then move the vertices five units up. To make the blade end in one vertex, scale the top vertices down to 0.000 (hold Ctrl for this) and press W>Remove Doubles (*Mesh Edit Menu*) or click the Rem Doubles button in the EditButtons (F9). Blender will inform you that it has removed seven of the eight vertices and only one vertex remains. The blade is complete! (*The completed blade*)

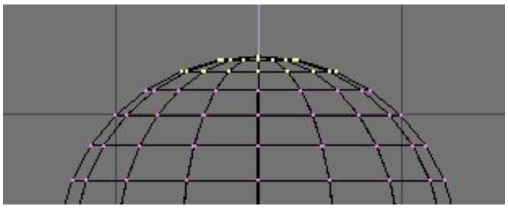
Mesh Edit Menu



The completed blade

The Handle

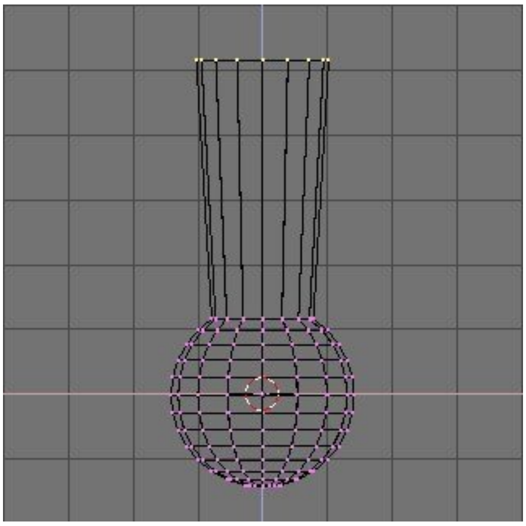
- Leave edit mode and move the blade to the side. Add a UVsphere with 16 segments and rings and deselect all the vertices with the A.



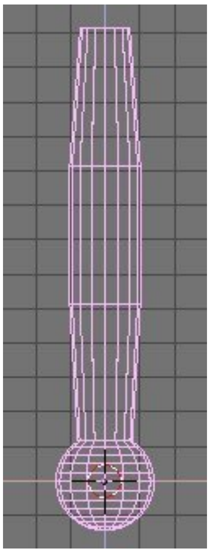
UV sphere for the handle: vertices to be removed

- Borderselect the top three rings of vertices with B and delete them with X>>Vertices (*UV sphere for the handle: vertices to be removed*).

- Select the top ring of vertices and extrude them. Move the ring up four units and scale them up a bit (*First extrusion for the handle*), then extrude and move four units again twice and scale the last ring down a bit (*Complete handle*).



First extrusion for the handle

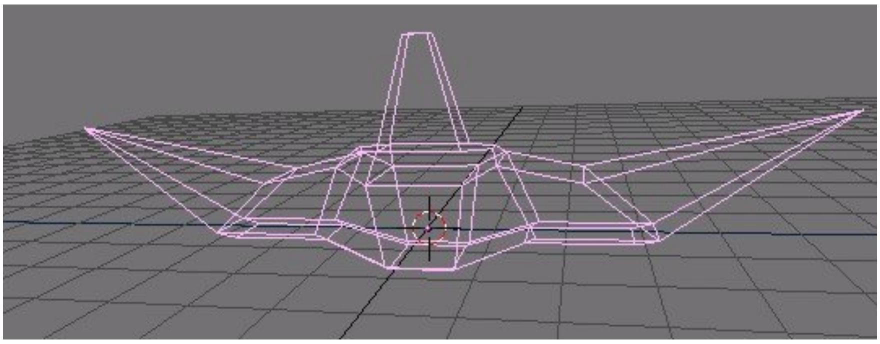


Complete handle

- Leave EditMode and scale the entire handle down so that it's in proportion with the blade. Place it just under the blade.

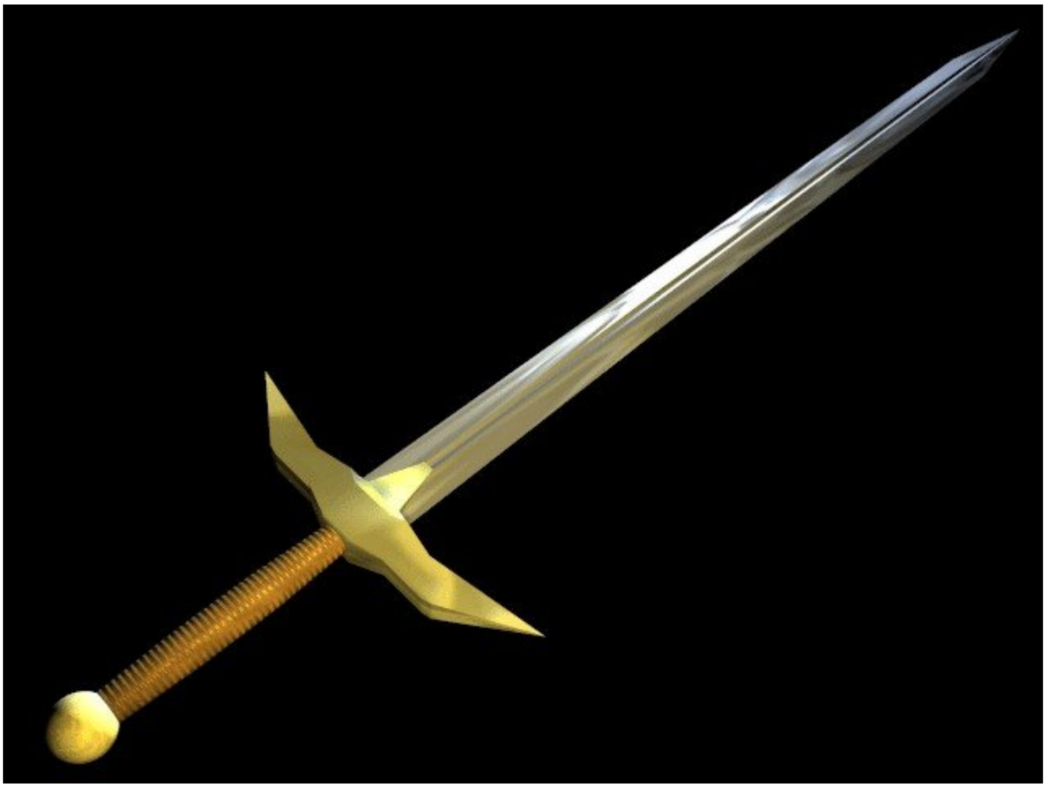
The Hilt

By now you should be used to the 'extrude>move>scale' sequence, so try to model a nice hilt with it. Start out with a cube and extrude different sides a few times, scaling them where needed. You should be able to get something like that shown in (*Complete Hilt*).



Complete Hilt

After texturing, the sword looks like (*Finished sword, with textures and materials*).



Finished sword, with textures and materials

As you can see, Extrude is a very powerful tool that allows you to model relatively complex objects very quickly; the entire sword was created in less than one half hour. Getting the hang of 'extrude>move>scale' will make your life as a Blender modeler a lot easier.

Mirror

Mode: Edit Mode, Object Mode

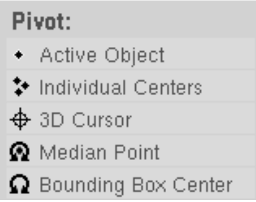
Hotkey: M in Edit Mode; Ctrl M in Object Mode

Menu: *In Edit Mode* :
geometry of the Object (mesh/curve/surface) â†’ Mirror â†’ Axis corresponding to the wanted transformation orientation
In Object Mode :
Object â†’ Mirror â†’ Local axis

The mirror tool is the exact equivalent of scaling by −1 to flip Objects, Vertice, Edges, Faces around one chosen pivot point and in the direction of one chosen axis only it is faster/handier.
Let's see this in detail.

In Edit Mode

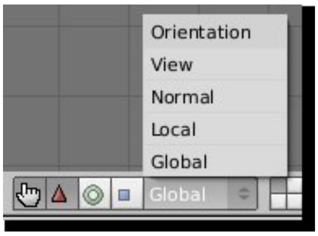
Pivot point



Pivot points

Pivot points must be set first. To learn more about the Pivot points see [this page](#)
Pivot points will become the center of symmetry.
If the widget is turned on it will always show where the Pivot point is.

Transformation orientation



The available tranform orientations for mirroring in Edit mode.

Transformation orientations are found on the 3D area header, next to the Widget buttons. They decide of which coordinate system will rule the mirroring. For mirroring the available transformation orientations are:

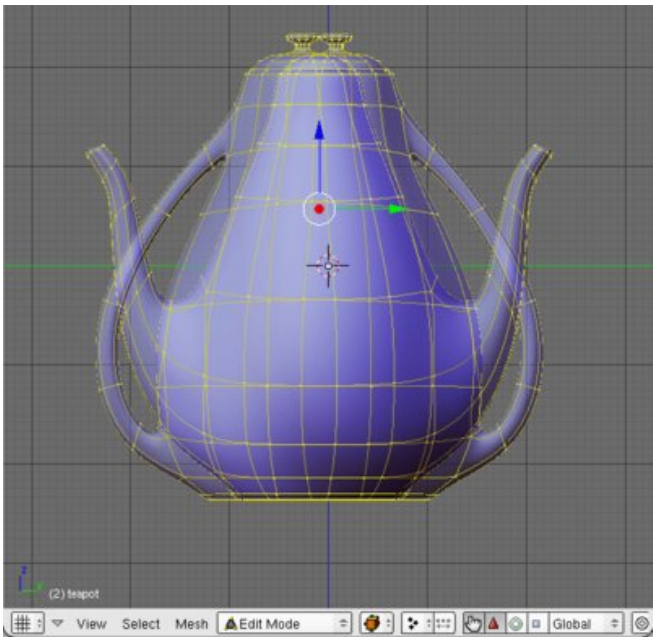
- View, i.e. the coordinate system of the view plane of the 3D area where the transformation will occur.
- Normal, i.e. the coordinate system based on the direction and location of normals for Meshes;
- Local, i.e. the coordinate system of the Object itself;
- Global, i.e. the coordinate system of the World;

Axis of symmetry

For each transformation orientation one symmetry axis

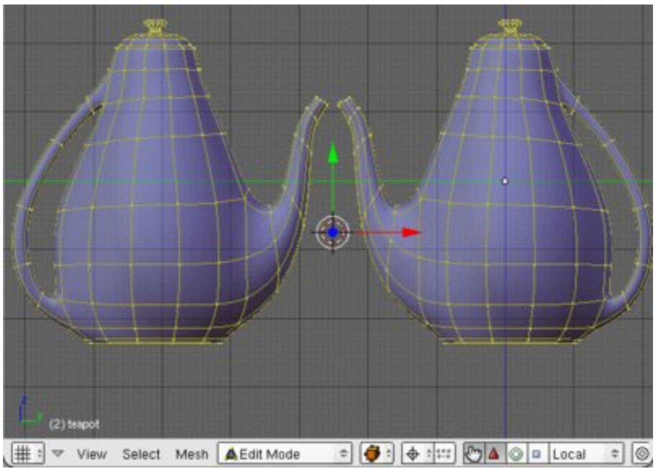
For each transformation orientation one of its axis along which the mirroring will occur.
As we can see the possibilities are infinite and the freedom complete: we can position the Pivot point at any location around which we want the mirroring to occur, chose one transformation orientation and then one axis

on it.
Here are three examples given to help figuring out what needs to be done and what result can be expected. In each case the whole geometry was duplicated with Shift D and the resulting copy was mirrored.



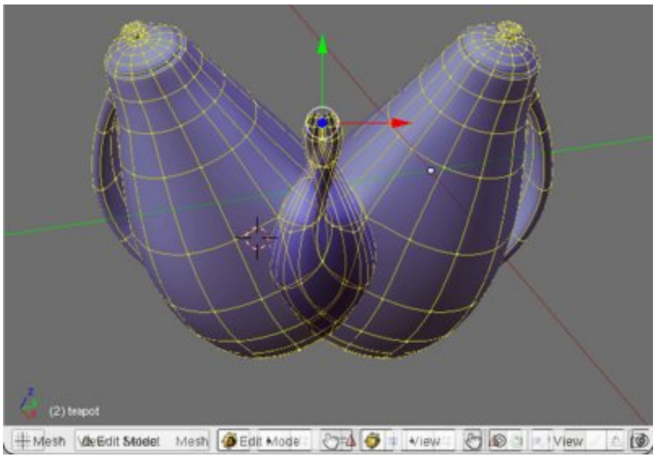
Mirror around the Individual center and along the Global Y axis.

On *Mirror around the Individual center...* the Pivot point default to **Median point of the selection of vertices** in Edit mode. This is a special case of the Edit mode as explained on the [Pivot point page](). The chosen transformation orientation is Global and the chosen axis is Y.



Mirror around the 3D Cursor and along the Local X axis.

On *Mirror around the 3D cursor...* the Pivot point is the 3D Cursor, the transformation orientation is Local, a.k.a. the Object space, and the axis of transformation is X.

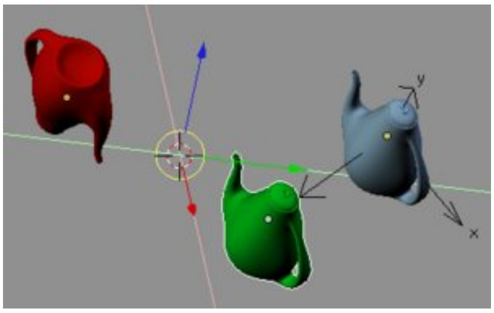


Mirror around an active vertex and along the View X axis

On *Mirror around an active vertex...* the vertex at the very tip of the spout is the Pivot point by choosing the *Active Object* option and then A to select all vertice followed by RMB twice on that vertex to make it active. The transformation orientation is View and the axis of transformation is X.

In Object Mode

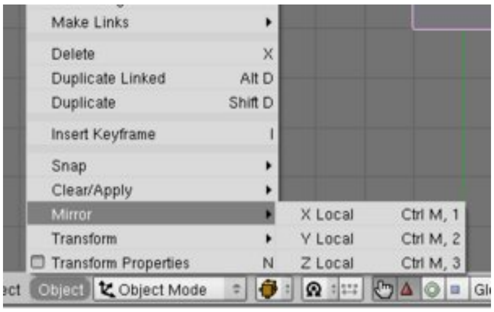
Mirroring is also available in Object mode but it is limited to Local (Object space) transformations. Any other orientation gives wrong results. The following example shows what could go wrong and what a proper result should look like.



Only Local space works for mirroring in Object space.

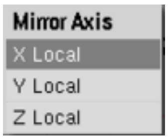
On *Only Local space works...* the red teapot is a mirrored copy of the blue one along the Global Y axis. Because the blue teapot is rotated relatively to the World this mirror resulted into an upside down copy of the original. Any transformation that is made at an angle from the Local axes of the transformed object will give wrong results.

The green teapot is also a copy of the blue one but is has been mirrored along the Local Z axis of that same blue teapot, resulting in a perfect mirror copy (colors were added afterward).



Mirror menu in Object mode.

On *Mirror menu in Object mode* we can see the choice of the three Local axes which are the only ones usable. Notice also the shortcuts to the extreme right.



Pop−up menu after using Ctrl M

On *Pop−up menu...* we see the same choices which pop up in the 3D area after using the Ctrl M shortcut.

In Conclusion

To summarize our survey of the Mirror tool a few recommendations.
Do not mistake it for the Mirror modifier.
Also remember that the results are the exact equivalent of a scale=−1 along an axis of the current transform orientation (TO). The advantage of the Mirror tool over this is that it is faster to use and almost foolproof.

Spin and SpinDup

Spin and Spin Dup are two very powerful modelling tools allowing you to easily create bodies of revolution or axially periodic structures.

Spin

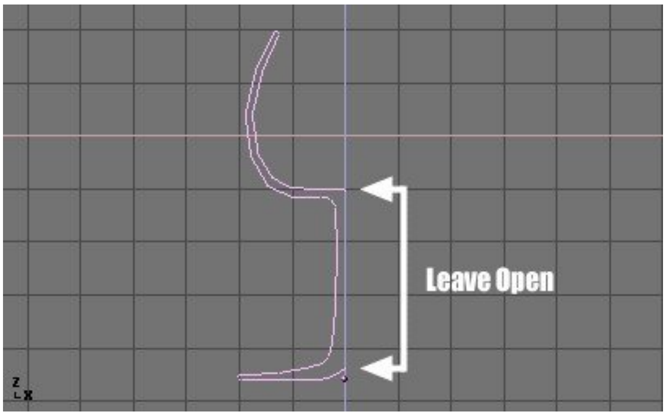
Mode: Edit Mode Editing context F9

Panel: Mesh Tools â†’ Spin

Use the Spin tool to create the sort of objects that you would produce on a lathe. (This tool is often called a

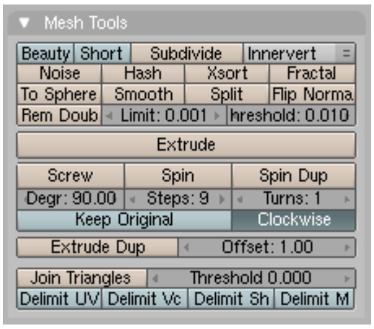
"lathe"−tool or a "sweep"−tool in the literature, for this reason.)

First, create a mesh representing the profile of your object. If you are modeling a hollow object, it is a good idea to thicken the outline. (*Glass profile*) shows the profile for a wine glass we will model as a demonstration.



Glass profile

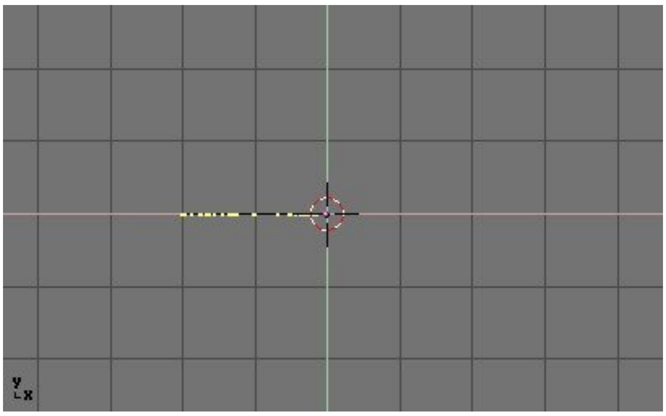
In EditMode, with all the vertices selected, access the Editing Context (F9). The Degr button in the [Mesh Tools](#) panel indicates the number of degrees to spin the object (in this case we want a full 360° sweep).



Spin Buttons

The Steps button specifies how many profiles there will be in the sweep (*Spin Buttons*).

Like Spin Duplicate (discussed in the next section), the effects of Spin depend on the placement of the 3D cursor and which window (view) is active. We will be rotating the object around the cursor in the top view. Switch to the top view with NumPad 7.

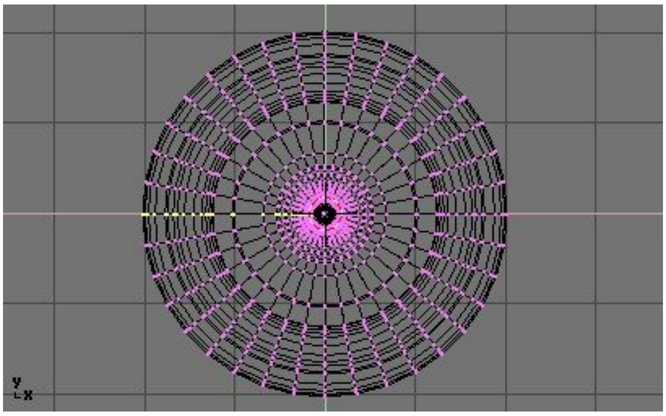


Glass profile, top view in edit mode, just before spinning.

- Place the cursor along the center of the profile by selecting one of the vertices along the center, and snapping the 3D cursor to that location with Shift S â†’ Cursor−>Selection (*Glass profile, top view in edit mode, just before spinning.*) shows the wine glass profile from top view, with the cursor correctly positioned.

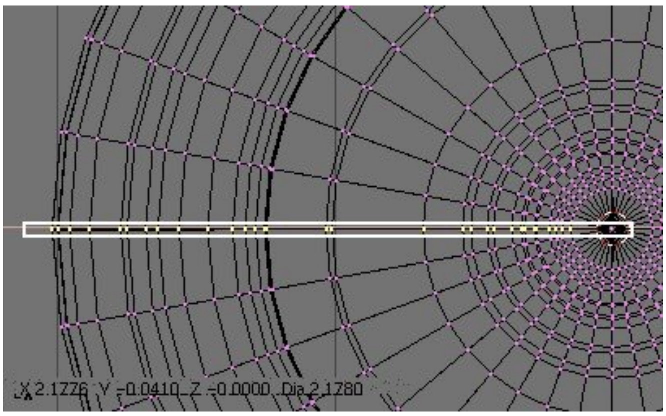
Mesh data − Vertex and face numbers.

Before continuing, note the number of vertices in the profile. You'll find this information in the Info bar at the top of the Blender interface (*Mesh data − Vertex and face numbers.*).



Spinned profile

- Click the "Spin" button. If you have more than one window open, the cursor will change to an arrow with a question mark and you will have to click in the window containing the top view before continuing. If you have only one window open, the spin will happen immediately. *Spinned profile* shows the result of a successful spin.



Seam vertex selection

- The spin operation leaves duplicate vertices along the profile. You can select all vertices at the seam with Box select (B) shown in (*Seam vertex selection*) and perform a *Remove Doubles* operation.

Notice the selected vertex count before and after the *Remove Doubles* operation (*Vertex count after removing doubles.*). If all goes well, the final vertex count (38 in this example) should match the number of the original profile noted in *Mesh data − Vertex and face numbers.*. If not, some vertices were missed and you will need to weld them manually. Or, worse, too many vertices will have been merged.

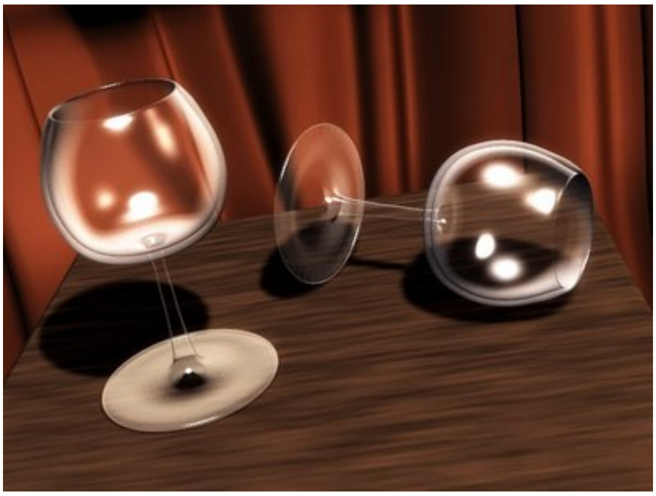
Vertex count after removing doubles.

Merge menu

Merging two vertices in one

To merge (weld) two vertices together, select both of them by holding Shift and RMB on them. Press S to start scaling and hold down Ctrl while scaling to scale the points down to 0 units in the X,Y and Z axis. LMB to complete the scaling operation and click the Remove Doubles button in the EditButtons window (also available with W â†' Remove Doubles). Alternatively, you can press W and select Merge from the appearing Menu (*Merge menu*). Then, in a new menu, choose whether the merged vertex will have to be at the center of the selected vertices or at the 3D cursor. The first choice is better in our case.

All that remains now is to recalculate the normals by selecting all vertices and pressing Ctrl N and selecting Recalc Normals Outside from the pop−up menu. At this point you can leave EditMode and apply materials or smoothing, set up some lights, a camera and make a rendering. (*Final render of the glasses.*) shows our wine glass in a finished state.



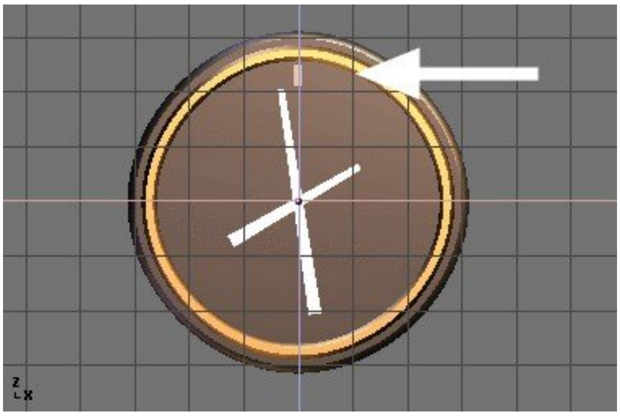
Final render of the glasses.

SpinDup

Mode: Edit Mode Editing context F9

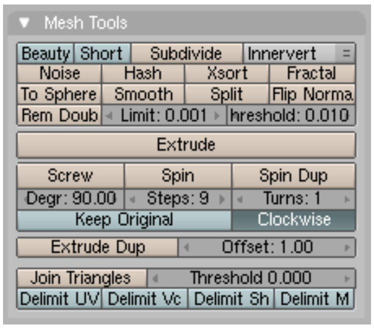
Panel: Mesh Tools Spin Dup

The Spin Dup tool is a great way to quickly make a series of copies of an object along a circle. For example, if you have modeled a clock, and you now want to add hour marks. Model just one mark, in the 12 o'clock position (*Hour mark indicated by the arrow*). Select the mark and switch to the Editing Context with F9.

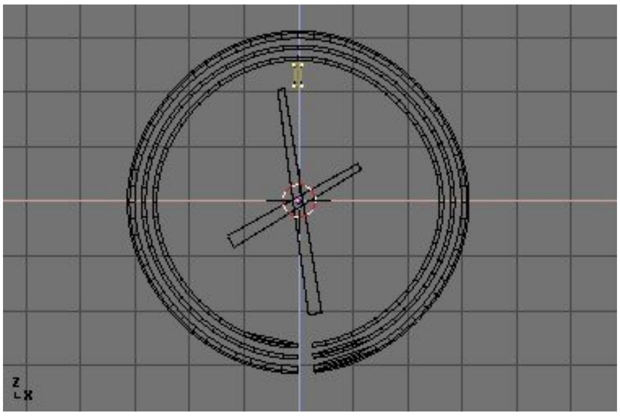


Hour mark indicated by the arrow

Set the number of degrees in the Degr: NumButton in the Mesh Tools panel to 360. We want to make 12 copies of our object, so set the Steps to 12 (*Spin Dup buttons*).



Spin Dup buttons

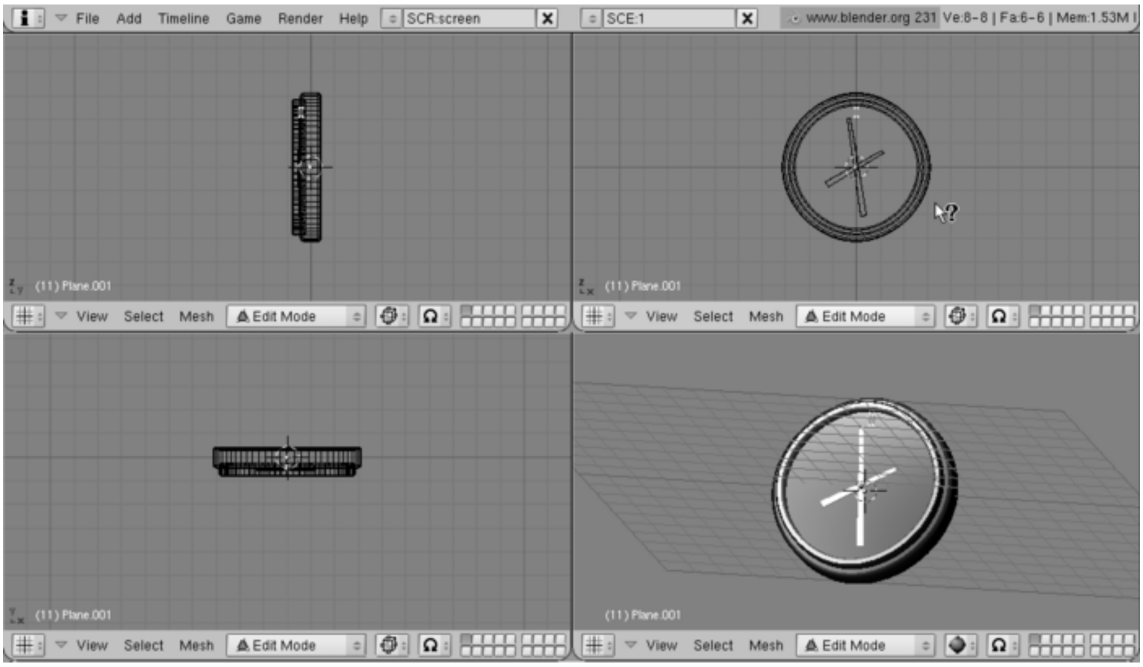


Mesh selected and ready to be SpinDuped

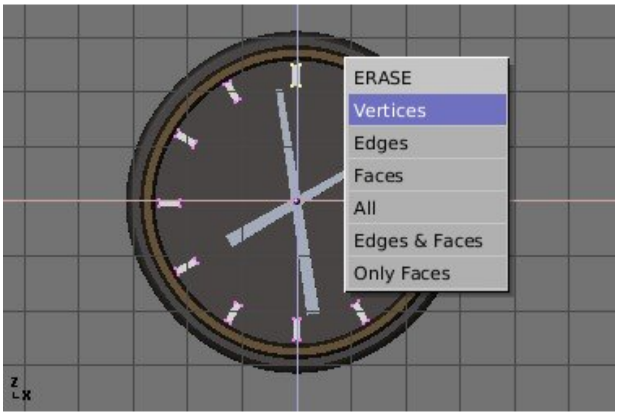
- Switch the view to the one in which you wish to rotate the object by using the keypad. Note that the result of the Spin Dup command depends on the view you are using when you press the button.
- Position the 3D cursor at the center of rotation. The objects will be rotated around this point. Note: To place the cursor at the precise location of an existing object or vertex, select the object or vertex, and press Shift S Cursor−>Selection.
- Select the object you wish to duplicate and enter *EditMode* with Tab.
- In *EditMode*, select the vertices you want to duplicate (note that you can select all vertices with A or all of the vertices linked to the point under the cursor with L). See (*Mesh selected and ready to be SpinDuped*).

- Press the Spin Dup button. If you have more than one 3DWindow open, you will notice the mouse cursor change to an arrow with a question mark. Click in the window in which you want to perform your rotation. In this case, we want to use the front window (*View selection for Spin Dup.*).

If the view you want is not visible, you can dismiss the arrow/question mark with Esc until you can switch a window to the appropriate view with the keypad.



View selection for Spin Dup.



Removal of duplicated object

When spin−duplicating an object 360 degrees, a duplicate object is placed at the same location of the first object, producing duplicate geometry.

You will notice that after clicking the Spin Dup button, the original geometry remains selected. To delete it, simply press X Vertices. The source object is deleted, but the duplicated version beneath it remains (*Removal of duplicated object*).

If you like a little math you needn't bother with duplicates because you can avoid them at the start. Just make 11 duplicates, not 12, and not around the whole 360°, but just through 330° (that is 360*11/12). This way no duplicate is placed over the original object.

In general, to make *n* duplicates over 360 degrees without overlapping, just spin one less object over 360*(*n*−1)/*n* degrees.

(*Final Clock Render*.) shows the final rendering of the clock.

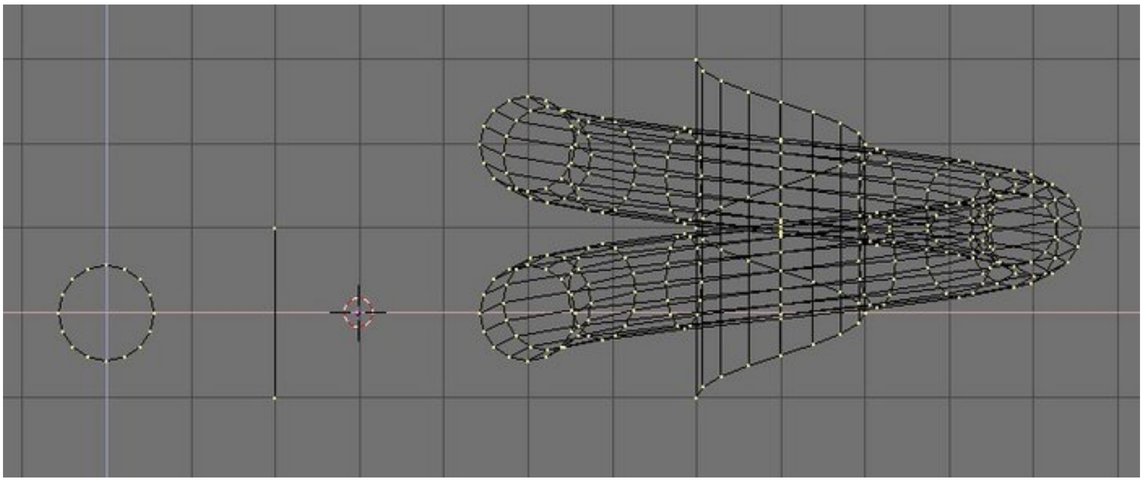
Final Clock Render.

Screw

Mode: Edit Mode Editing context F9

Panel: Mesh Tools Screw

The Screw tool combines a repetitive "Spin" with a translation, to generate a screw−like, or spiral−shaped, object. Use this tool to create screws, springs, or shell−shaped structures.



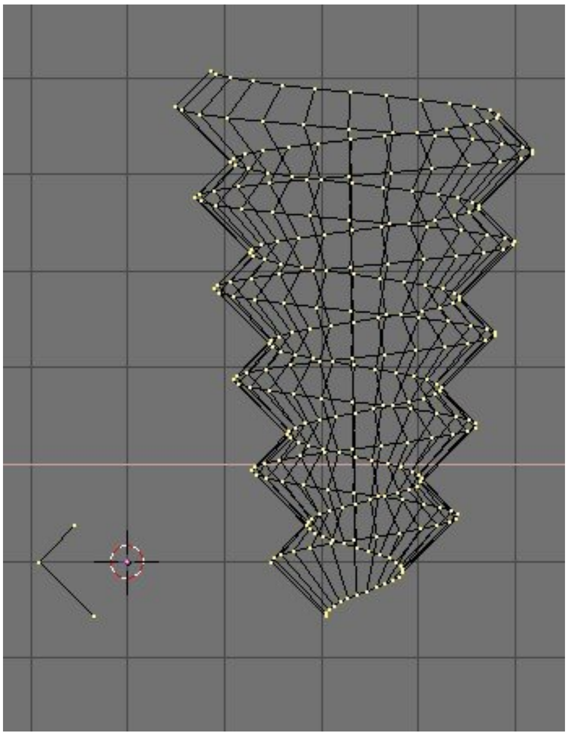
How to make a spring: before (left) and after (right) the Screw tool.

The method for using the "Screw" function is strict:

- Set the 3DWindow to *front view* (NumPad 1).
- Place the 3DCursor at the position through which the rotation axis must pass. The rotation axis will be vertical.
- Your mesh object must contain *both* the profile to be spun and an *open line of vertices* to define how the profile is translated as it is spun. In the simplest case, the open line also serves as the profile to be spun; alternatively, a separate closed line (e.g., a circle as shown in the figure) can be specified as the

profile. The open line can be a single edge, as shown in the figure, or a half circle, or whatever. You need only ensure that the line has two 'free' ends. (At a 'free' end, a vertex is connected to only one other vertex.) The Screw function uses these two points to calculate the translation vector that is added to the "Spin" for each full rotation (*How to make a spring: before (left) and after (right) the Screw tool.*). If these two vertices are at the same location, this creates a normal "Spin". Otherwise, interesting things happen!

- Select all vertices that will participate in the "Screw".
- Assign the NumButtons Steps: and Turns: in the Mesh Tools Panel the desired values. Steps: determines how many times the profile is repeated within each 360° rotation, while Turns: sets the number of complete 360° rotations to be performed.
- Press Screw



Enlarging screw (right) obtained with the profile on the left.

If there are multiple 3DWindows, the mouse cursor changes to a question mark. Click on the 3DWindow in which the Screw is to be executed.

If the two 'free' ends are aligned vertically, the result is as seen above. If they are not, the vertical component of the translation vector remains equal to the vertical component of the vector joining the two 'free' vertices, while the horizontal component generates an enlargement (or reduction) of the screw as shown in (*Enlarging screw (right) obtained with the profile on the left.*). (In this example the open line serves as the profile as well as defining the translation.)

Warp Tool

Mode: Edit Mode Editing context F9

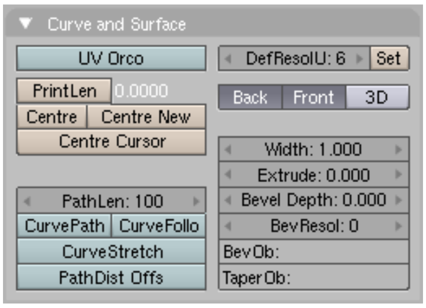
Panel: Mesh Tools Warp

The Warp tool is a little−known tool in Blender, partly because it is not found in the Edit Buttons window, and partly because it is only useful in very specific cases. At any rate, it is not something that the average

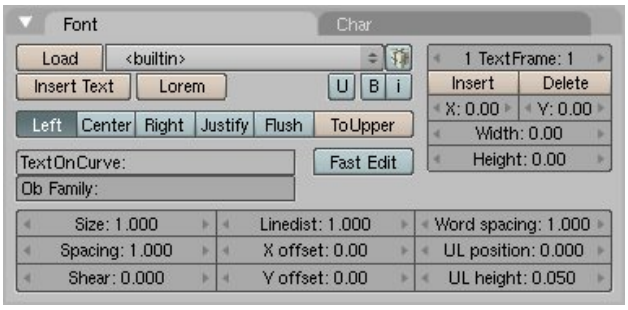
Blender−user needs to use every day.

A piece of text wrapped into a ring shape is useful when creating flying logos, but it would be difficult to model without the use of the warp tool. For our example, we'll warp the phrase "Amazingly Warped Text" around a sphere.

- First add the sphere.
- Then add the text in front view, in the Editing Context and Curve and Surface Panel set Extrude to 0.1 − making the text 3D, and set Bevel Depth to 0.01, adding a nice bevel to the edge. Make the Bev Resol 1 or 2 to have a smooth bevel and lower the resolution so that the vertex count will not be too high when you subdivide the object later on using (*Curve and Surface* and *Font*) panels.
- Convert the object to curves, then to a mesh, (Alt C twice) because the warp tool does not work on text or on curves.
- Subdivide the mesh twice (W Subdivide Multi 2), so that the geometry will change shape cleanly, without artifacts.

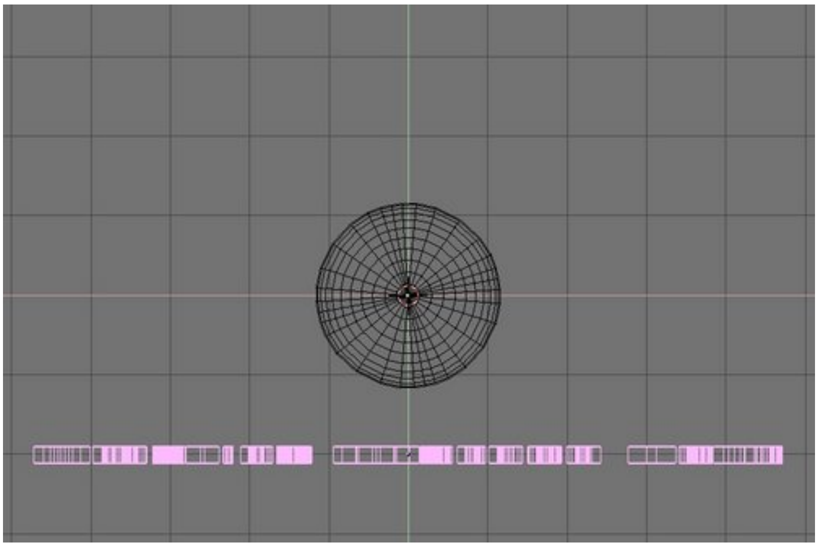


Curve and Surface.



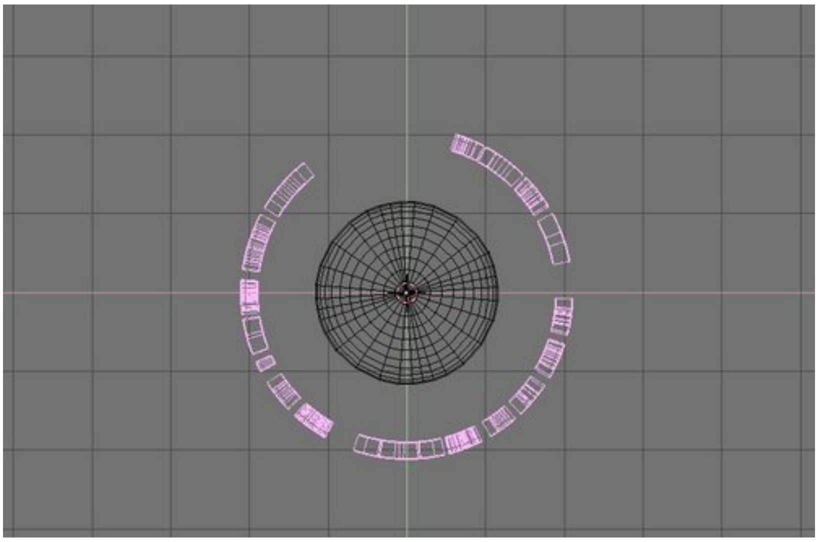
Font.

Switch to top view and move the mesh away from the 3D cursor. This distance defines the radius of the warp. (See *Top view of text and sphere*.)



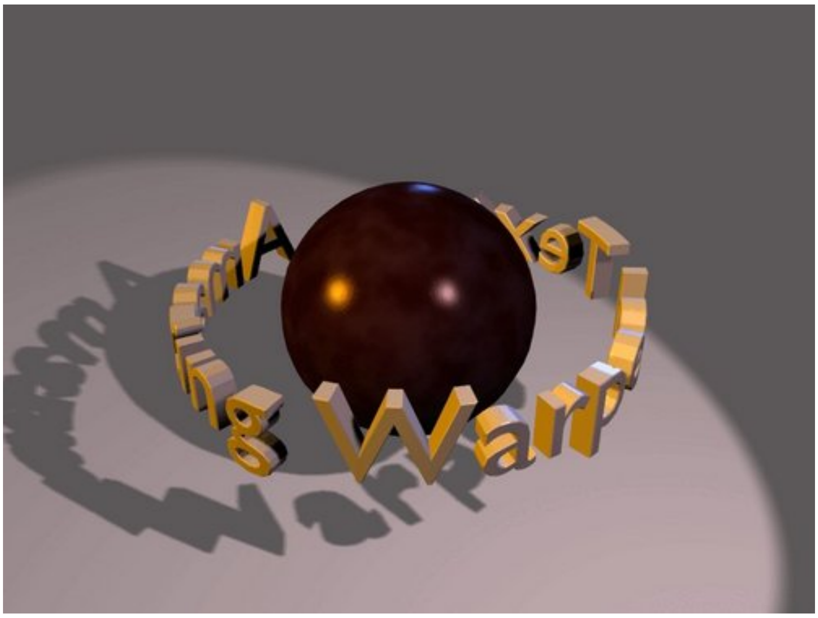
Top view of text and sphere

Place the mesh in Edit Mode (Tab) and press A to select all vertices. Press Shift W to activate the warp tool. Move the mouse up or down to interactively define the amount of warp. (*Warped text*). Holding down Ctrl makes warp change in steps of five degrees.



Warped text

Now you can switch to camera view, add materials, lights and render (*Final rendering*).



Final rendering

To Sphere

Mode: Edit Mode Editing context F9

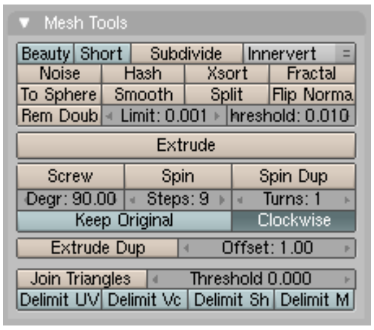
Panel: [Mesh Tools](#) To Sphere

Hotkey: Ctrl Shift S

Another of the lesser known tools is To Sphere (Ctrl Shift S). This command allows the creation of spheres from subdivided cubes.

First, start with a Cube. I will start with from fresh by Erasing All (Ctrl X).

- Press Tab to switch into Edit Mode.



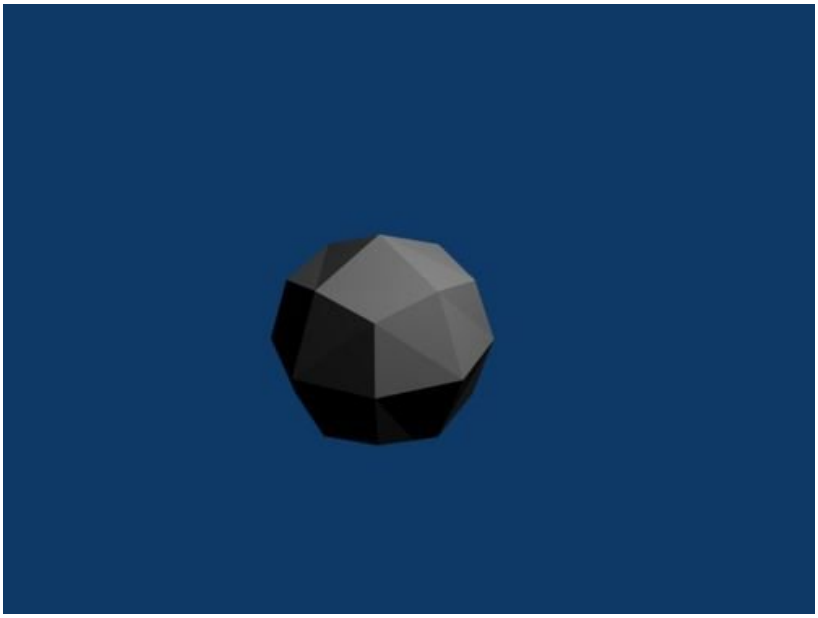
Extrude button in EditButtons context.

- Make sure all the vertices of the cube are selected by pressing A twice. Then, go to the Editing Panel by pressing F9. You should be able to see the Mesh Tools section now.

- Subdivide the cube by pressing the Subdivide button in the Mesh Tools section, or by pressing W and clicking "Subdivide". You can do this as many time as you want; the more you subdivide, the smoother your sphere will be.

- Click the To Sphere button now in the Mesh Tools. Select "100" to make your sphere. Alternatively, you can press Ctrl Shift S and type in "1.000" to achieve the same effect.

The completed sphere!



Finished low−res sphere!

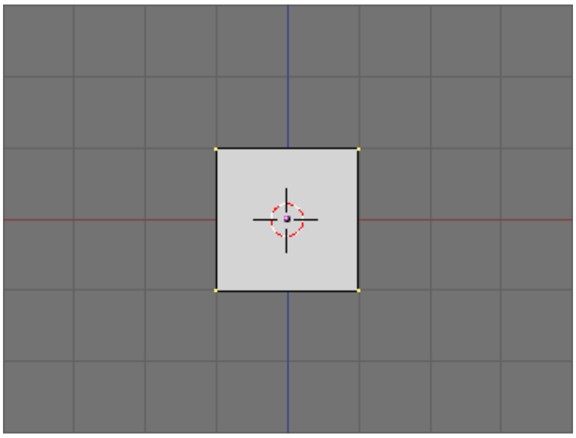
Previous: Manual/Selecting Meshes	Contents	*Next:* Manual/Advanced Mesh Tools

User Manual: Contents | Guidelines | Blender Version **2.4**

Editor's Note: this is more of a tutorial than a user manual, but we don't have anything else at the moment that covers the topic.−−Roger 02:17, 29 May 2007 (CEST)

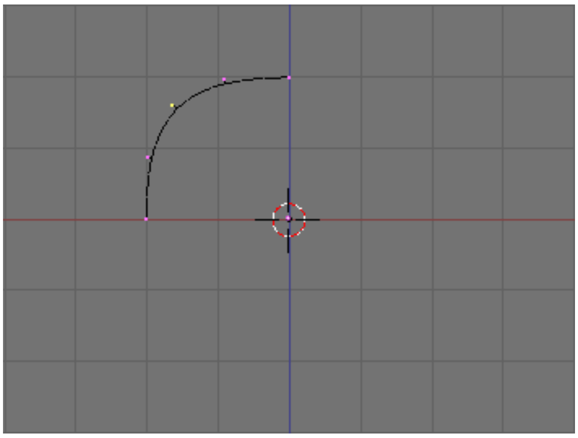
Advanced Mesh Modelling

Symmetrical Modelling



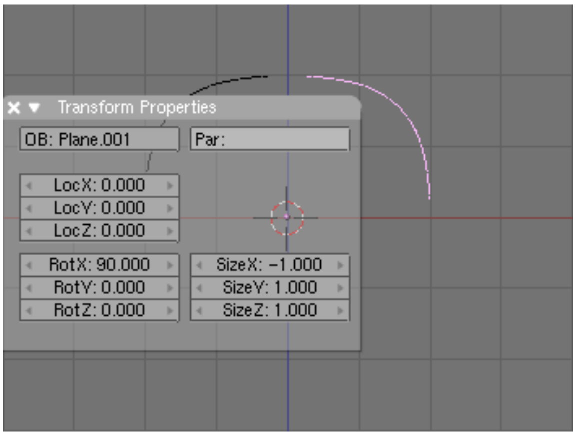
A plane

You often need to model objects which exhibit some sort of symmetry. For radial, rotational or multiple symmetry the best approach is to carefully model one base structure and then, as a last step, duplicate the base cell via SpinDup or whichever command is most appropriate. For objects with bilateral symmetry, those with one plane of symmetry, such as most animals (humans included) and many machines, the above method implies modelling one half of the object, and then mirroring a duplicate of the first half to get the whole object. Since it is usually difficult to attain correct proportions by only modelling a half, it is possible to duplicate the half before it is completely modelled, and act on one half and automatically update the other.



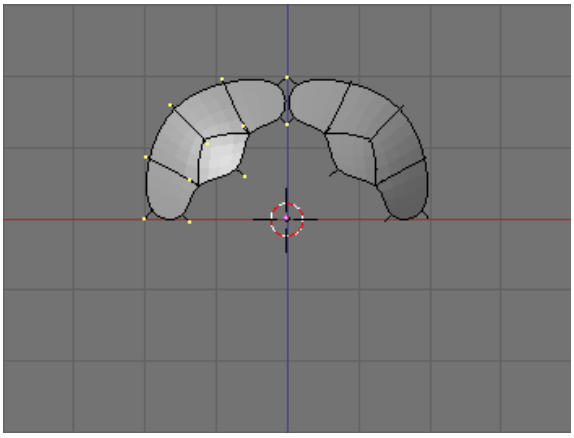
Right half.

In Front View add a plane or whatever (*A plane*.). Consider it as a starting point for one half of the object. Let's say the object's right half, which for us in frontal view is on the left of the screen. The plane of symmetry is the yz plane. Move the mesh, in EditMode, so that it is completely on the left of center. Delete some nodes, and add some others, to give it its general shape, as in *Right half.*.



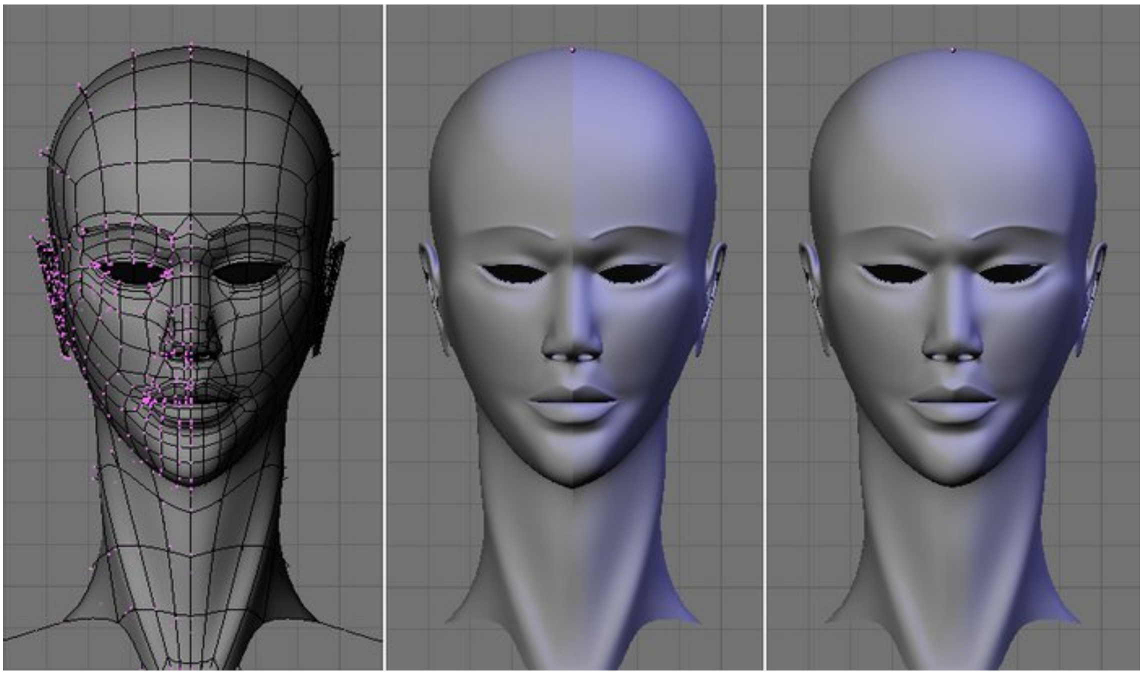
Mirroring the linked duplicate.

Now switch to ObjectMode and, with the half selected, make a *linked* duplicate with Alt D. Press ESC to exit from Grab Mode and press N. In the Numeric input panel which appears, set SizeX to `-1` (*Mirroring the linked duplicate.*). This effectively mirrors the linked duplicate with respect to the Object's center, hence the importance of keeping the center on the plane of symmetry.



Editing one half.

Having duplicated the Object as a linked duplicate implies that the two objects share the *same* mesh data, which is implicitly mirrored by the unitary negative scaling along the x axis, which is normal to the symmetry plane. Now you can edit either of the two halves. Since they share mesh data any change, be it an extrude, delete, face loop cut etc... immediately reflects on the other side (*Editing one half.*).
By carefully editing one half, and possibly by using a blueprint as a background to provide guidelines, very interesting results can be achieved (*A head. Left: EditMode; Center: ObjectMode; Right: Joined.*, left).



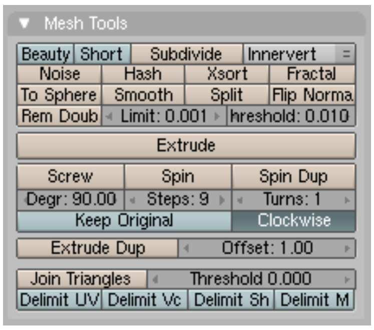
A head. Left: EditMode; Center: ObjectMode; Right: Joined.

As a final step, when symmetrical modelling is complete, the two halves must be selected and joined into a single Object (Ctrl J). This makes the seam (very visible in *A head. Left: EditMode; Center: ObjectMode; Right: Joined.*, center) disappear. Once you have a single object (*A head. Left: EditMode; Center: ObjectMode; Right: Joined.*, right), you can start modelling the subtle asymmetries which every being has.

Note

In Blender 2.33 and earlier versions the OpenGL implementation causes mirrored linked duplicates to have wrong normals, so that one of the two halves is black. This is fixed in Blender 2.34, but older versions can use this technique anyway by setting the mesh to single sided while symmetrical modelling is used.

Noise



Noise button in EditButtons

The Noise function allows you to displace vertices in meshes based on the grey−values of a texture applied to it. So, you have to have a texture assigned to the material, even if that texture is not Mapped To anything. In

your texture, you should enable *No RGB* to convert color textures to a gradient. You should also have subdivided your object enough to have many vertices to act on.

Use Noise to generate great landscapes or make mesh surfaces more real−world (pitted, un−smooth). The Noise function displaces vertices in the object's Z−Axis and negative Z−Axis only. To deform your mesh's other dimensions, simply rotate your object and *apply rotation*, or rotate the vertices in edit mode, and apply Noise. Then, rotate it back again to get your original orientation.

Noise permanently modifies your mesh according to the material texture. Each click adds onto the current mesh. For a temporary effect, map the texture to *Disp*lacement for a render−time effect. In object/edit mode your object will appear normal, but will render deformed.

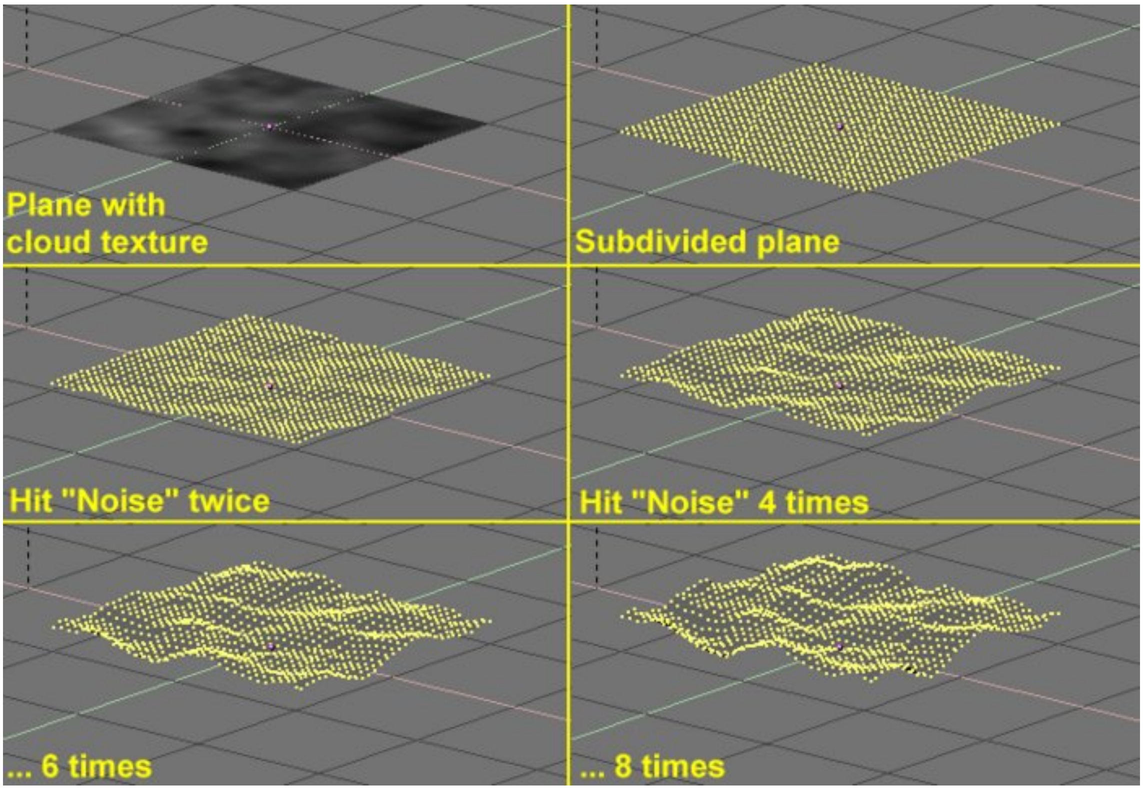
Example

Subdivide tool

Add a plane and subdivide it at least five times. To do that you can either use the Subdivide or Subdivide Multi entry in the *Specials* menu accessed via W; see (*Subdivide Tool*). Using Subdivide Multi is faster and easier. Select Subdivide Multi and enter `5` for the number of cuts popup dialog.

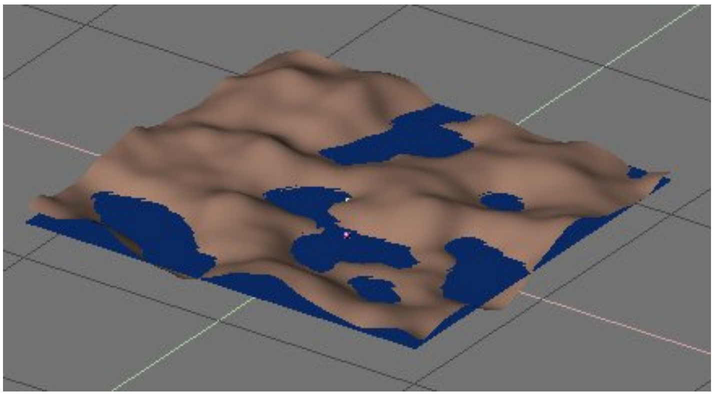
Now add a material and assign a Clouds texture to it. Adjust the NoiseSize: to `0.500`. Choose white as the color for the material and black as the texture color, to give us good contrast for the noise operation.

Ensure that you are in EditMode and that all vertices are selected, then switch to the Editing Context F9. Press the Noise button in the Mesh Tools Panel (*Noise button in EditButtons*) several times until the landscape looks nice. (*Noise application process*) is an example of applying the noise tool. From top left to bottom right: Plane with texture, sub−divided plane, "Noise" button hit 2, 4, 6 and 8 times. *shows the original − textured − plane as well as what happens as you press Noise.*



Noise application process.

Remove the texture from the landscape now because it will disturb the look. Then add some lights, some water, smooth the terrain, and so on. (*Noise generated landscape*).



Noise generated landscape

Note

The noise displacement always occurs along the mesh's *z* coordinate, which is along the direction of the *z* axis of the Object local reference.

Previous: Manual/Basic Mesh Tools	Contents	*Next:* Manual/Edge and Face Tools

User Manual: Contents | Guidelines | Blender Version **2.4**

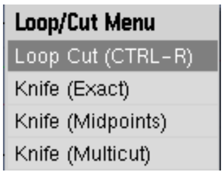
Edge and Face Tools

Mode: Edit Mode (Mesh)

Hotkey: Ctrl E or K

Menu: Mesh Edges

Description

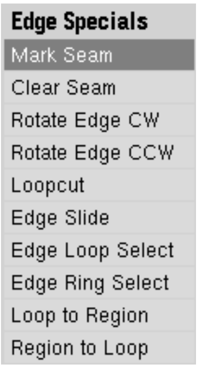


Loop/Cut Menu

A key issue in modelling is the necessity to add vertices in certain zones of the mesh, and this is often regarded as splitting, or adding, edges in a given region.

Loops often play an important role in this process. For a brief introduction to Loops please refer to Edge and Face Loops.

Many Edge Tools are grouped in a menu which is linked to K hotkey (*Loop/Cut Menu*), but each individual tool has its own hotkey as well.



Edge Specials Menu

Edge Tools for selection and manipulation are grouped in a menu which is linked to Ctrl E hotkey (*Edge Specials Menu*).

Edge Slide, for example, slides the vertices in the loop along the edges. If you select a loop on an egg−shaped object, for example, sliding the vertices will move them, not left/right or up/down, but instead proportionally move them as if they were sliding along the edge using the edge as a guide. More information on selecting loops is found below.

Tools don't work on Modifiers

In general, you cannot use any tool on a mirrored side, as that side is just a mirror image of the primary side. Tools also do not work on subsurfed or multires "edges" shown; use the tool by working on vertices/edges/faces on the primary part of your mesh when using a modifier.

Edge Selection

Mode: Edit Mode (Mesh)

Hotkey: RMB

Description

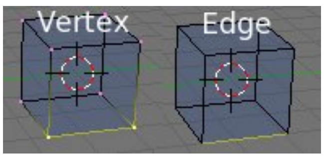
Select modes.

In Edit Mode there are a few ways to select edges: implicitly, explicitly, looping or by Region Selection. *Implicit* means you describe a more complex element using less complex elements. For example, to describe an edge you need to specify two vertices and to describe a face you need to specify three or more vertices or three or more edges.

Region Selection is a tool that allows selection of edges and faces based on an intersection with a rectangular, circular or lasso 2D region.

Explicit Edge Selection
: To select an edge use edge select mode and RMB click on an edge. To select additional edges use Shift RMB.

Implicit Edge Selection



Implicit Edge selection.

> The other way to select an edge is to select two vertices that bound the edge of interest. You are implying which edge by selecting its bounding vertices. To select an edge implicitly use vertex select mode in combination with RMB and/or Shift RMB.
>
> In (*Implicit Edge selection*), the cube on the left shows an edge selected using vertices. The cube on the right is what shows when you switch to edge select mode.

Options

If you are in solid, shaded, or textured viewport shading mode (not bounding box or wireframe), you will have a fourth button that looks like a cube. When enabled, this limits your ability to select based on visible edges (as if the object was solid), and prevents you from accidentally selecting, moving, deleting or otherwise working on backside or hidden edges.

Edge Loop Selection

Mode: Edit Mode (Mesh) > Vertex Select Mode or Edge Select Mode

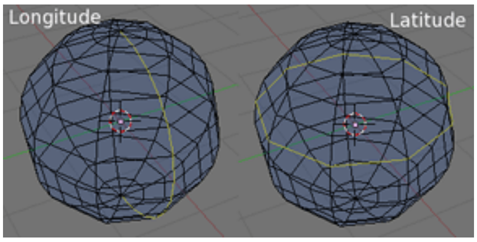
Hotkey: Alt RMB or Ctrl E, 7 (based on existing edge selection)

Menu: Select Edge Loop (based on existing edge selection)

Description

Holding Alt while selecting an edge selects a loop of edges that are connected in a line end to end, passing through the edge under the mouse pointer. Holding Shift while clicking adds to the current selection. Edge loops can also be selected based on an existing edge selection, using â€˜Edge Selectâ€™ in the Edge Specials menu.

Examples



Longitudinal and latitudinal edge loops

The left sphere shows an edge that was selected longitudinally. Notice how the loop is open. This is because the algorithm hit the vertices at the poles and terminated because the edge at the pole connects to more than 3 other edges. However, the right sphere shows an edge that was selected latitudinally and has formed a closed loop. This is because the algorithm hit the first edge that it started with.

Technical Details

The algorithm for selection is as follows:

- First check to see if the selected element connects to only 3 other edges.
- If the edge in question has already been added to the list, the selection ends.
- Of the 3 edges that connect to the current edge, the ones that share a face with the current edge are eliminated and the remaining edge is added to the list and is made the current edge.

Loop to Region and Region to Loop

Mode: [Edit Mode](#) (Mesh)

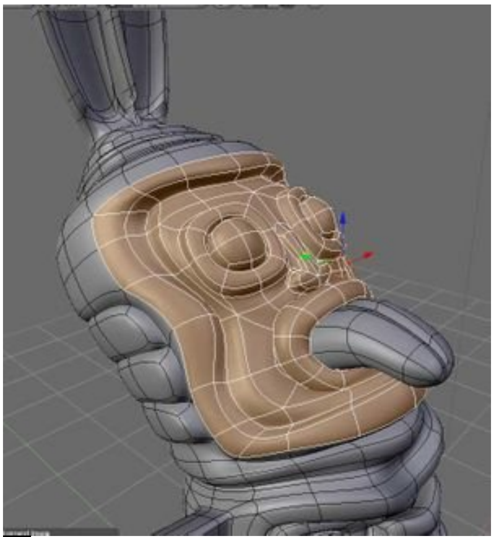
Hotkey: Ctrl E, 9 and Ctrl E, 0

Menu: Select Loop to Region, Region to Loop

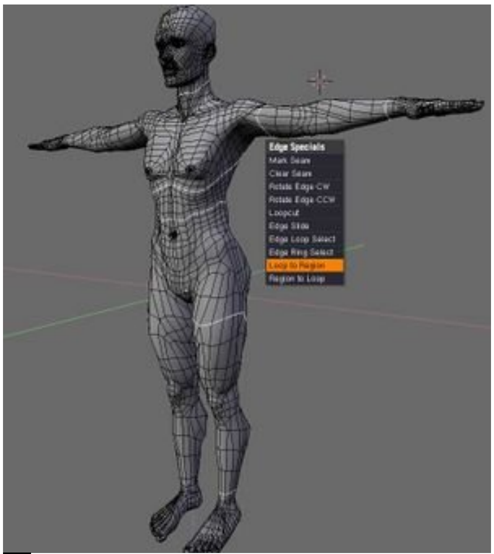
Description

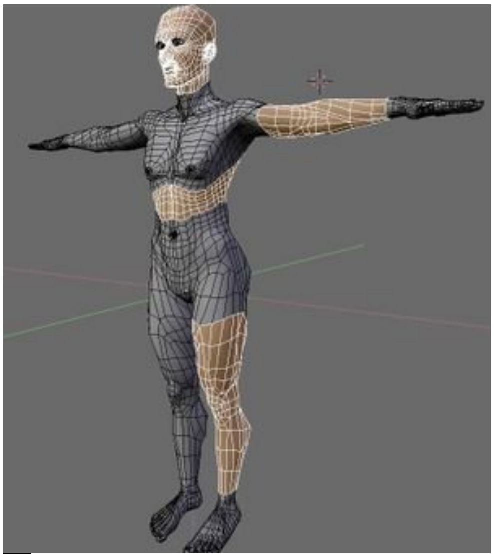
Examines the current set of selected edges and seperates them into groups of 'loops' that each bisect the mesh into two parts. Then for each loop it selects the smaller 'half' of the mesh.

Examples − Loop to Region:



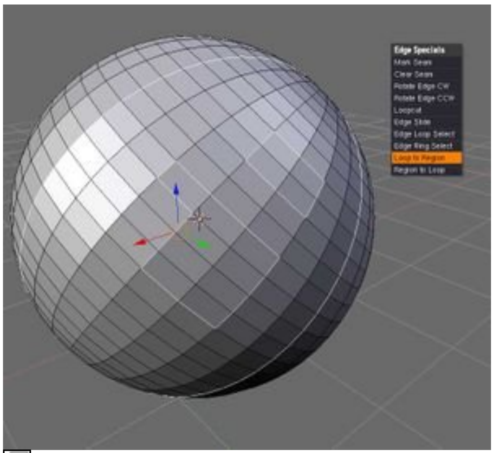
Selection Loop to Region

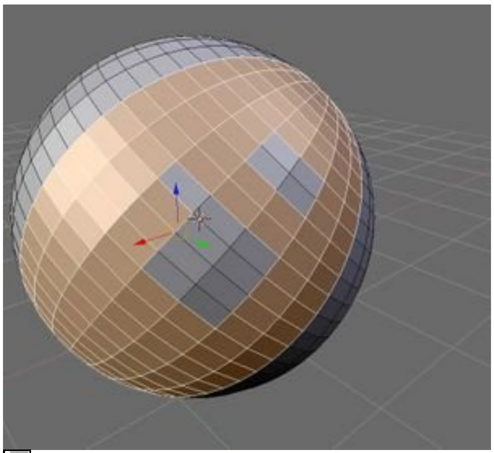




Selection

This tool handles multiple loops fine as you can see.

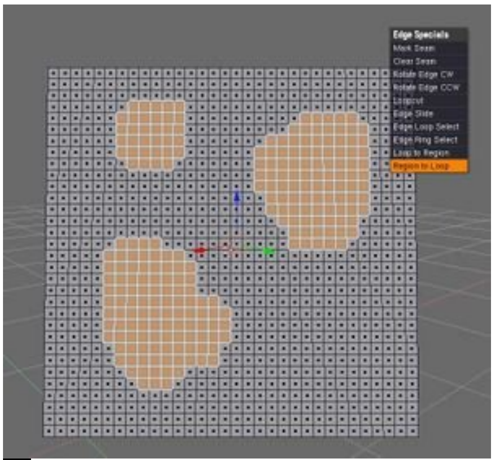


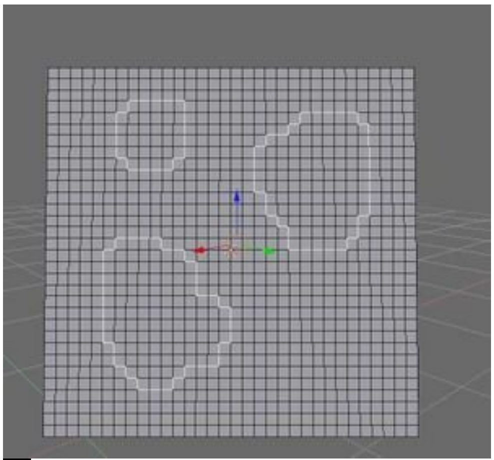


Selection

This tool handles 'holes' just fine as well.

Example − Region to Loop:





Selection

This is the 'logical inverse' of loop to region.

Edge Ring Selection

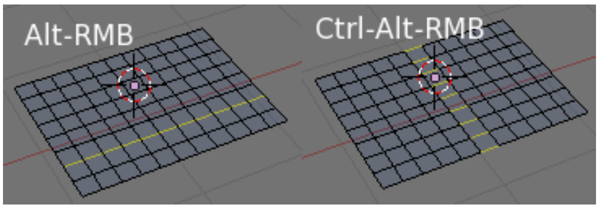
Mode: [Edit Mode](#) (Mesh) > Edge Select Mode

Hotkey: CtrlAlt RMB

Description

In edge select mode, holding Ctrl Alt while selecting an edge selects a sequence of edges that are not connected, but on opposite sides to each other continuing along a face loop. Using the same command in vertex select mode will select such a face loop implicitly.

Examples



A selected edge loop, and a selected edge ring

In (*A selected edge loop, and a selected edge ring*), the same edge was clicked on but two different "groups of edges" were selected, based on the different commands. One is based on edges during computation and the other is based on faces.

Technical Details

Edge ring selection is based on the same algorithm as in [Face Loop Selection], though the end results differ as only edges are selected.

Face Selection

Mode: Edit Mode (Mesh)

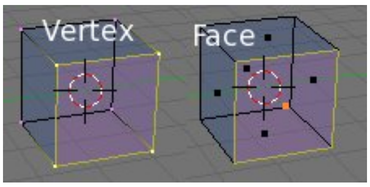
Hotkey: RMB

Description

In *EditMode* there are a few ways to select **Faces**: implicitly, explicitly, looping or by region. *Implicit* means you describe a more complex element using less complex elements. For example, to describe an edge you need to specify two vertices and to describe a face you need to specify three or more vertices or three or more edges.

Explicit Face Selection
: To select a face use *Face select mode* (*Select modes.*) and the RMB. To select additional faces use Shift RMB.

Implicit Face Selection



Implicit Face selection.

Selecting the three or four vertices that bound the face of interest in vertex select mode selects it implicitly. (*Implicit Face selection*) shows a face selected on a cube using vertices the cube of the left. The cube on the right is what shows when you switch to Face select mode.

You can also implicitly select faces by selecting the bounding edges of the face of interest. This will produce

the same results as selecting vertices.

Deleting Faces may delete edges and vertices

If a vertex that defines a selected face is not connected to anything else, and you delete the face, Blender deletes the vertex as well as its connecting edges. This is so you don't end up with a bunch of stray vertices running around unconnected in 3D space. If you want a vertex and/or an edge to stay, you have first to see it (vertices + edges are both visible in vertex select mode; only the edges are visible in edge select mode; neither the vertices or the edges are visible in face select mode). Then you must use X > Faces Only or X > Edges and Faces, accordingly to what you want to keep of what you see.

Options

If you are in solid, shaded, or textured viewport shading mode (not bounding box or wireframe), you will have a fourth button that looks like a cube. When enabled, this limits your ability to select from visible faces (as if the object was solid), and prevents you from accidentally selecting, moving, deleting or otherwise working on backside or hidden faces.

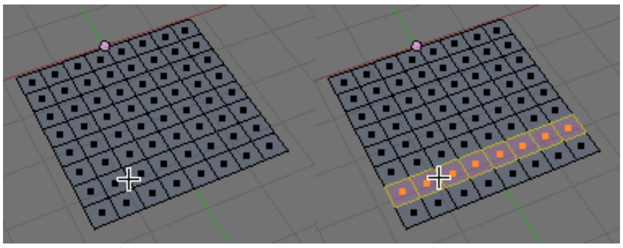
Face Loop Selection

Mode: Edit Mode (Mesh) > Face Select Mode or Vertex Select Mode

Hotkey: Alt RMB (face select mode) or Ctrl Alt RMB (vertex select mode)

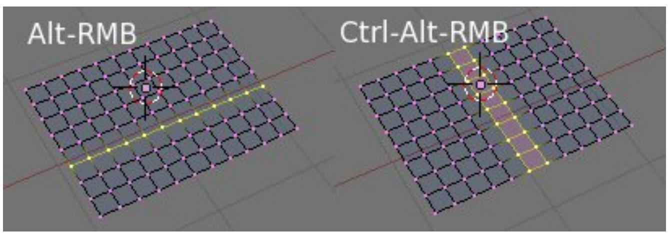
In face select mode, holding Alt while selecting an edge selects a loop of faces that are connected in a line end to end, along their opposite edges. In vertex select mode, the same can be accomplished by using Ctrl Alt to select an edge ring, which selects the face loop implicitly.

Examples



Face loop selection

This face loop was selected by clicking with Alt RMB on an edge, in face select mode. The loop extends perpendicular from the edge that was selected.



Alt versus Ctrl−Alt in Vertex Mode.

A face loop can also be selected in Vertex select mode, see (*Alt versus Ctrl−Alt in Vertex Mode*). The edges selected in the grid labeled "Alt−RMB" is a result of selecting and edge loop versus selecting an edge ring. Because in Vertex Select Mode, selecting opposite edges of a face implicitly selects the entire face, the face loop has been selected implicitly.

Note that in these cases the generated result of the algorithm was *vertices* because we were in Vertex select mode. However, had we had been in Edge select mode the generated result would have been selected edges.

Technical Details

The algorithm for selection is as follows:

- A face loop is made by two neighbouring edge loops.
- Extends only to quadrilateral faces.
- Ends when a triangular face is met (and the two bounding edgeloops merge into one).

Loop Subdivide

Mode: Edit Mode (Mesh)

Hotkey: Ctrl R

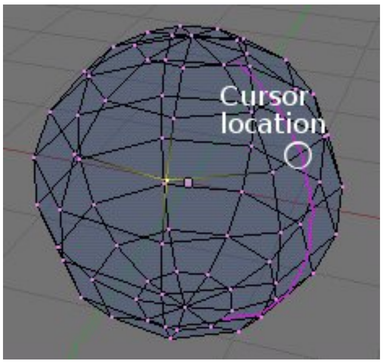
Menu: Mesh Edges Loop Subdivide...

Description

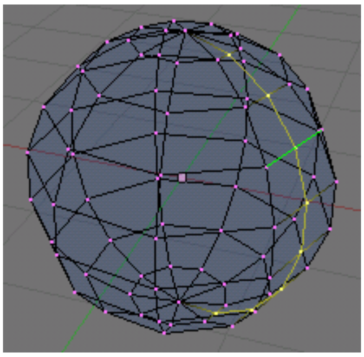
Loop Subdivide splits a loop of faces by inserting a new edge loop intersecting the chosen edge. The tool is interactive and has two steps:

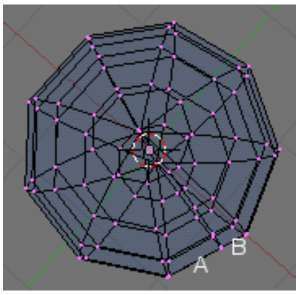
1. Previsualising the cut
: The cut to be made is marked with a magenta coloured line as you move the mouse over the various edges. In (*Pre−visualising the cut*), the mouse cursor was located where the white circle is located. This caused the loop line to appear at the mid point of the edge. The to−be−created edge loop stops at the poles where the existing face loop terminates.

2. Sliding the new edge loop
: Once an edge is chosen via LMB, that edge is highlighted in green (*Loop Split edge selected*) and you can move the mouse along the edge to determine where the new edge loop will be placed. This is identical to the Edge Slide tool. Clicking LMB again confirms and makes the cut at the pre−visualised location, or clicking MMB forces the cut to exactly 50%.
(*Loop Split edge completed*) shows the new faces and edges, **A** and **B**. The view is rotated so that the new faces and edges are clearly visible from the top of the sphere.



Pre−visualising the cut

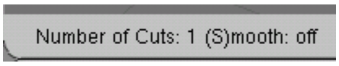

2. Sliding the new edge loop



3. Loop Split edge completed.

Options

1. Previsualising the cut



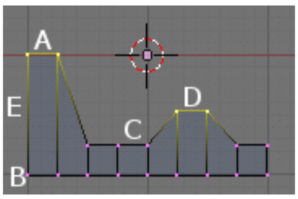
Pre−visualising (header)

- Upon initial activation of the tool 3D window header changes to show the "Number of Cuts" (*Initial Loop Split header*). Entering a number with the keyboard, scrolling MW or using NumPad + and NumPad − changes the number of cuts (maximum of 130). These cuts run parallel with the face loop line.
- S changes the cut to 'smooth' mode. By default, new vertices for the new edge loop are placed exactly on the pre−existing edges. This keeps subdivided faces flat, but can distort geometry, particularly when using Subdivision Surfaces. If smooth mode is on then new vertices are not placed on the previous edge but shifted outwards by a given percentage, similar to the 'Subdivide Smooth' command.

2. Sliding the new edge loop

- P switches between *Proportional* and *Percentage* modes. The default mode is *Percentage*.
- In *Proportional* mode, MW , or â† and â†’ changes the selected edge for calculating a proportion.
- Holding Ctrl or SHIFT control the precision of the sliding. Ctrl snaps movement to 10% steps per move and Shift snaps movement to 1% steps. The default is 5% steps per move.

Examples



Loop Example Grid.

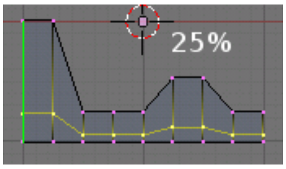
In order to explain *Proportional* and *Percentage* modes we can use a very simple mesh layed out like a 2x9 grid, (*Loop Example Grid*). The vertices at **A** and **D** have been moved in order to emphasize the difference between the two modes. The vertices at the level **C** and **B** remain unchanged. **E** is an area of interest when looking at *Proportional* mode.

Percentage mode

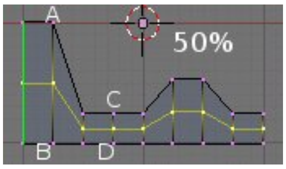
Percentage header.

In *Percentage* mode the 3D window header changes to (*Percentage header*) showing a number between **−1** and **1** with **0** representing 50% or midway between.

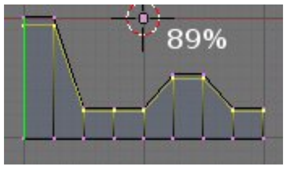
As you move the mouse the percentage changes and the edge loop line, drawn in yellow, moves as a percentage of the distance between the edge marked in green as shown in (*25% between*), (*Mid−way*) and (*89% between*).



25% between



Mid−way



89% between

The yellow loop line is *always* the same percentage along edge of the edges that are being cut, regardless of the edges' lengths. For example, in (*Mid−way*) the yellow loop line is exactly halfway between vertex **A** and **B** and it is also exactly halfway between vertex **C** and **D**. For (*25% between*) you can see that the yellow line loop is *always* 25% along each of the cut edges.

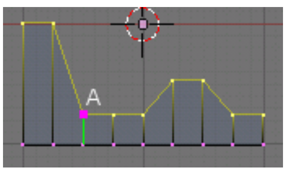
Proportional mode

Proportional header.

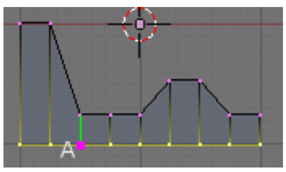
Proportional face loop splitting keeps the shape of the newly cut edge loop the same as one of the edge loops that it cuts between, rather than cutting at a percentage along each perpendicular edge.

In *Proportional* mode the 3D window header changes to (*Proportional header*) showing the position along the length of the currently selected edge which is marked in green. Movement of the sliding edge loop is restricted to this length. As you move the mouse the length indicator in the header changes showing where along the length of the edge you are.

Unlike *Percentage* mode, *Proportional* mode treats the edge as having a start and end vertex with the start marked by a magenta marker (*Vertex Marker*). The start vertex (**A**), can be flipped to the opposite vertex using F (*Vertex Marker Opposite*).



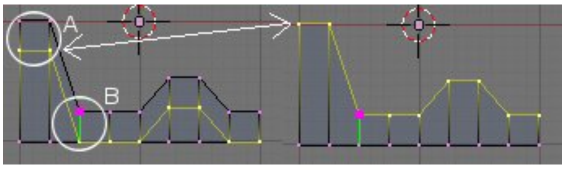
Vertex Marker.



Vertex Marker Opposite.

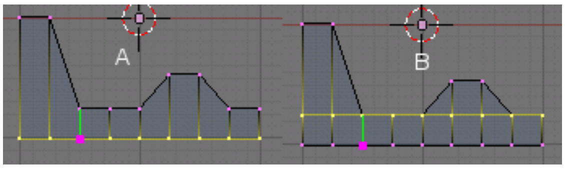
Moving the mouse moves the cut line towards or away from the start vertex, but the loop line will only move as far as the length of the currently selected edge, conforming to the shape of one of the bounding edge loops.

(*Proportional Range*) shows an example of how the distance is restricted by the length of the current edge (**B**). Looking at (**A**), you can see that the line loop has moved the same distance. If the line only moves 0.2 units on the selected edge then the line only moves 0.2 units everywhere else in the face loop region. The portion of the line loop at **A** hasn't gone all the way to the "bottom" because the selected edge is only 0.25 units in length. The line portion at "**A**" will not be able to move more than 0.25 units down because the range of movement is restricted to the length of the selected edge.



Proportional Range

(*Proportional Range Flipped*) is another example where the start vertex has been flipped while using the same selected edge as compared to (*Proportional Range*). You can see that movement is still restricted to the length of the selected edge. The yellow edge loop line stays straight, conforming to the bottom bounding edge loop because the cut is placed a constant distance from the bottom edge loop, along the crossed edges.



Proportional Range Flipped

Delete Edge Loop

Mode: Edit Mode (Mesh)

Hotkey: X or Delete

Menu: Mesh Edges Delete Edge Loop

Description

Erase Menu.

Delete Edge Loop allows you to delete a selected edge loop if it is between two other edge loops. This will create one face−loop where two previously existed.

Note

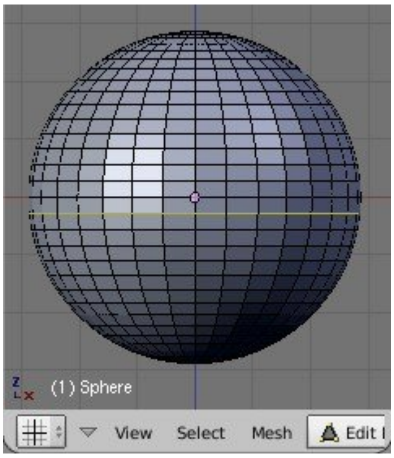
The Edge Loop option is very different to the Edges option, even if you use it on edges that look like an edge loop. Deleting an edge loop merges the surrounding faces together to preserve the surface of the mesh. By deleting a chain of edges, the edges are removed, deleting the surrounding faces as well. This will leave holes in the mesh where the faces once were.

Limitations & Work−arounds

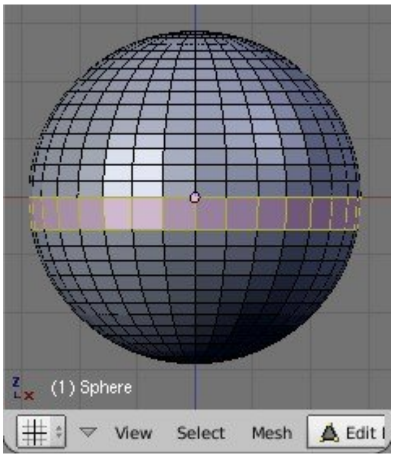
For Edge Loop Delete to work correctly, a single edge loop must be selected. The same restrictions as those of Edge Slide apply, see Edge Slide for more details.

Examples

The selected edge loop on the UV Sphere has been deleted and the faces have been merged with the surrounding edges. If the edges had been deleted by choosing "Edges" from the (*Erase Menu*) there would be an empty band of deleted faces all the way around the sphere instead.



Before Delete Edge Loop



After Delete Edge Loop

Knife Subdivide

Mode: Edit Mode (Mesh)

Panel: Editing Context Mesh Tools

Hotkey: K or Shift K

Menu: Mesh Edges Knife Subdivide...

Description

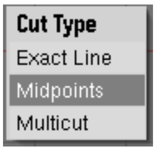
Knife Subdivide subdivides **selected** edges intersected by a user−drawn 'knife' line. For example, if you wish to cut a hole in the front of a sphere, you select only the front edges, and then draw a line over the selected

edges with the mouse. The tool is interactive, and only works with primary edges; selected either implicitly by selecting all or explicitly by box−selecting or shift−selecting a few edges.

Tools don't work on Modifiers

In general, you cannot use any tool on a mirrored side, as that side is just a mirror image of the primary side. Tools also do not work on subsurfed or multires "edges" shown; use the tool by working on vertices/edges/faces on the primary part of your mesh when using a modifier.

Options



Knife Tool/Cut Type

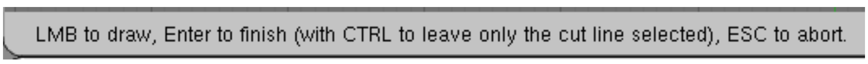
When you press K, the popup menu appears where you select the type of cut you can make:

- Exact Line divides the edges exactly where the knife line crosses them.
- Midpoints divides an intersected edge at its midpoint.
- Multicut makes multiple parallel cuts. An additional number input is presented, allowing you to select the number of cuts.

Drawing the cut line
: When using Knife Subdivide, the cursor changes to an icon of a scalpel and the 3D View header shows (*Knife Tool 3DWindow header*). You can draw connected straight lines by clicking LMB and moving repeatedly or you can create *freehand* lines by pressing and holding LMB while dragging. Also, exact cuts on the vertices can be made by holding Ctrl while cutting. MMB constrains the cut line to a vertical or horizontal screen axis.

Confirming and selection
: Pressing Esc or RMB at any time cancels the tool, and pressing Enter confirms the cut, with the following options:

- Enter will leave selected every edge except the new edges created from the cut.
- Ctrl Enter will select only the new edges created from the cut. Note: only edges that intersect the hand−drawn selected edges will be selected.



Knife Tool 3DWindow header

Topology
: Knife subdivide uses the same options as the other subdivide tools, located in the Edit Buttons. If the Beauty option is toggled selected faces are only subdivided along the longest 2 sides. If both Beauty and Short options are toggled, selected faces are only subdivided along the shortest 2 sides.

Note: Using edge select mode to select only the edges you wish to subdivide creates a more accurate subdivision than using the "Beauty" toggle.

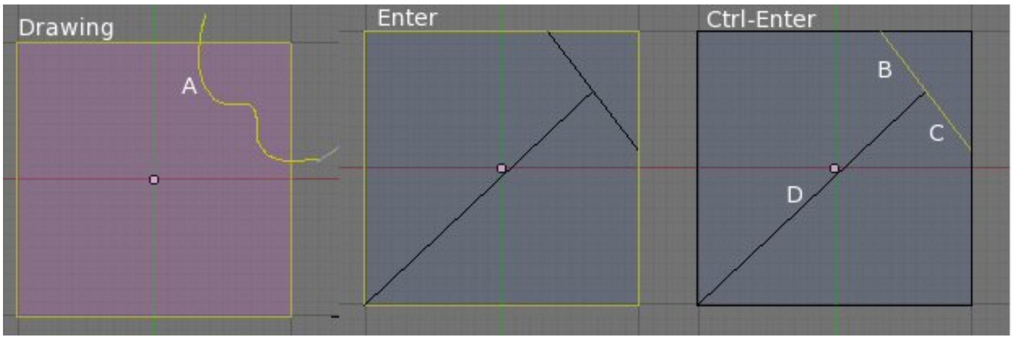
Examples

Exact Line Cut Type

(*Exact Line before and after*) is an example of using the 'Exact Line' knife. The cut is determined from the hand−drawn line labeled **A** in the plane labeled "**Drawing**".

The plane labeled "**Enter**" is the result of hitting the Enter key. The intersections on the edges of the plane are where the drawn line actually intersects, no matter how wiggly the line is. In addition, all the edges have been selected other than the newly created edges from the cut tool itself.

The plane labeled "**Ctrl−Enter**" is the result of hitting Ctrl Enter. In this case only the newly created edges, **B** and **C**, are selected while edge **D** is not. **D** is a secondary edge added as a side effect of the cut tool.



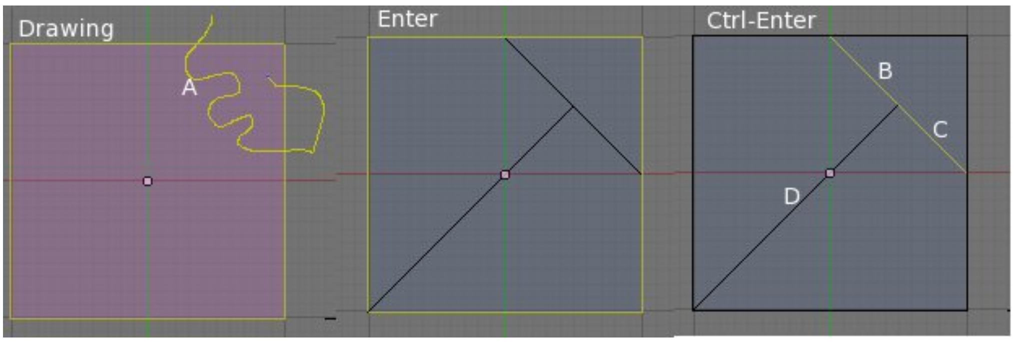
Exact Line, before and after.

Midpoints Cut Type

(*Midpoints before and after*) is an example of using the 'Midpoints Line' knife. The cut is determined from the hand−drawn line labeled **A** on the plane labeled "**Drawing**". Notice how the line labeled **A** intersects the right edge twice; only the first intersection will be considered during the cut.

The plane labeled "**Enter**" is the result of hitting Enter. The intersections on the edges of the plane are the mid points of each edge, regardless of where the line was drawn. All the edges have been selected other than the newly created edges from the cut tool itself.

The plane labeled "**Ctrl−Enter**" is the result of hitting Ctrl Enter. In this case only the newly created edges, **B** and **C**, are selected while edge **D** is not. **D** is a secondary edge as a result of the cut tool.



Midpoints before and after.

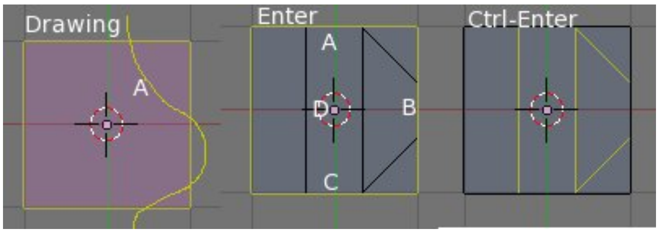
MultiCut Type

Number of Cuts.

This cut type presents a popup dialog appears asking for the **Number of Cuts**, which defines how many equally spaced cuts the tool should make for each intersecting edge. For example, the default of "**2**" generates two intersections or three new edges for each intersection of the hand−drawn line.

(*MultiCut before and after*) is an example of using the 'MultiCut' knife. The cut is determined from the hand−drawn line (**A**) on the plane labeled "**Drawing**", while using the default "2" as the number of cuts. The line was drawn so that it deliberately intersected three edges.

The grid labeled "**Enter**" is the result of hitting Enter. There are two cuts equally spaced on each edge intersected by the hand−drawn line; labeled **A**, **B** and **C**. **D** is a secondary edge as a result of the cut tool. The grid labeled "**Ctrl−Enter**" is the result of hitting Ctrl Enter. In this case only the newly created edges are selected while edge **D** is not.



MultiCut before and after.

Limitations & Work−arounds

The cut lines can be drawn with any number of segments, but only one intersection is detected one crossing per edge. Crossing back over an edge multiple times does not perform additional cuts on it.

Snap to grid is not currently implemented, but is being looked at for future releases.

Optimizations

With a large mesh, it will be quicker to select a smaller number of vertices, those defining only the edges you plan to split since the Knife will save time in testing selected vertices for knife trail crossings.

Rotate Edge CW / Rotate Edge CCW

Mode: Edit Mode (Mesh)

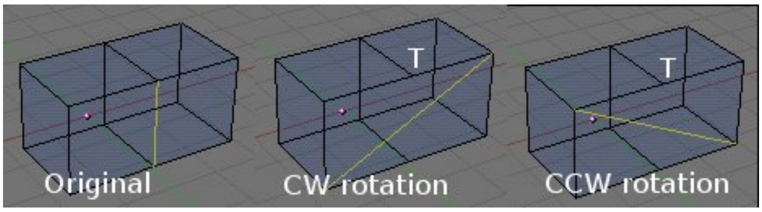
Hotkey: Ctrl E, 3 and Ctrl E, 4

Menu: Mesh Edges Rotate Edge CW / Rotate Edge CCW

Description

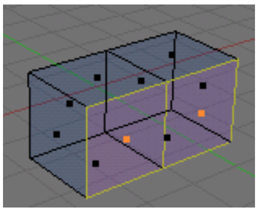
Rotating an edge clockwise or counter−clockwise spins an edge between two faces around their vertices. This is very useful for restructuring a mesh's topology. The tool can operate on one explicitly selected edge, or on two selected vertices or two selected faces that implicitly select an edge between them.

Examples



Selected Edge Rotated CW and CCW.

Be aware that sometimes, as shown in (*Selected Edge Rotated CW and CCW*, indicated with a **T**, that you could produce what appears to be "T" junctions/nodes by using this tool. However, Blender has created additional edges that prevent cracks in the mesh. You can test this by selecting the vertex at the **T** and moving it around while noting that there are two edges now instead of one long edge.



Adjacent selected Faces

To rotate an edge based on faces you must to select two faces, (*Adjacent selected Faces*), otherwise Blender notifies you with an error dialog, "ERROR: Select one edge or two adjacent faces." Using either Rotate Edge CW or Rotate Edge CCW will produce exactly the same results as if you had selected the common edge shown in (*Selected Edge Rotated CW and CCW.*).

Edge Slide

Mode: Edit Mode (Mesh) > Vertex Select Mode or Edge Select Mode

Hotkey: Ctrl E, 6

Menu: Mesh Edges Slide Edge

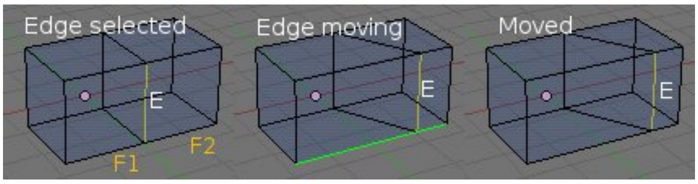
Description

Edge Slide slides one or more edges along faces adjacent to the selected edge(s) with a few restrictions involving the selection of edges.

Options

- LMB confirms the tool, and RMB or Esc cancels
- As with [Loop Subdivide], this tool has both a *Percentage* and *Proportional* mode, which is displayed in the 3D View header. These modes behave the same as in [Loop Subdivide] including all keys for controlling precision edge movement.

Examples



Simple edge slide

(*Simple edge slide*) is an example of sliding an edge along an extruded box. The selected edge is labeled **E** and the adjacent faces to that edge are **F1** and **F2**. In **Edge moving**, the edge is being slid along the edge drawn in green. **Moved** shows the results.

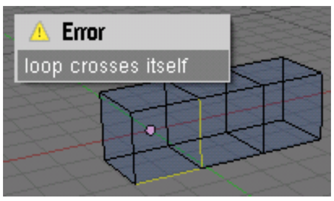
Limitations & Work−arounds

There are restrictions on the type of edge selections that can be operated upon. Invalid selections are:

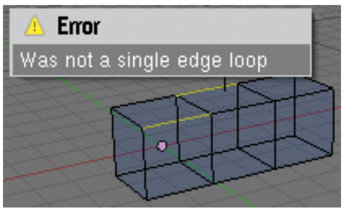
Loop crosses itself
: This means that the tool could not find any suitable faces that were adjacent to the selected edge(s). (*Loop Crosses*) is an example that shows this by selecting two edges that share the same face. A face can not be adjacent to itself.

Was not a single edge loop
: Most likely you have selected edges that don't share the same edge loop. (*Single Edges*) is an example where the selected edges are not in the same edge loop which means they don't have a common edge. You can minimize this error by always selecting edges end to end or in a "Chain".

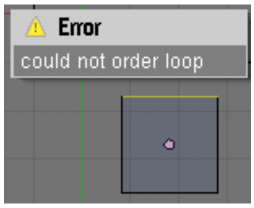
Could not order loop
: This means the tool could not find an edge loop based on the selected edge(s). (*Order loop*) is an example where a single edge was selected in a 2D *Plane* object. An edge loop can not be found because there is only one face. Remember, edge loops are loops that span two or more faces.



Loop Crosses.



Single Edges.



Order loop.

A general rule of thumb is that if multiple edges are selected they should be connected end to end such that they form a continuous chain. This is *literally* a general rule because you can still select edges in a chain that are invalid because some of the edges in the chain are in different edge loops. (*Loop Crosses*) is just such an example where the selected edges form a chain but they are not in the same edge loop.

If you select multiple edges just make sure they are connected. This will decrease the possibility of getting looping errors.

Bevel

Mode: Edit Mode (Mesh)

Hotkey: W, Alt 2

Menu: Mesh Edges Bevel

Description

With bevel and without bevel

A bevel is something that smooths out a sharp edge or corner. True world edges are very seldom exactly sharp. Not even a knife blade edge can be considered perfectly sharp. Most edges are intentionally bevelled for mechanical and practical reasons.

Bevels are also useful for giving realism to non−organic models. In the real world, the blunt edges on objects catch the light and change the shading around the edges. This gives a solid, realistic look, as opposed to un−bevelled objects looking false.

Options

Recursion

The number of recursions in the bevel can be defined in an additional popup number field. The greater the number of recursions, the smoother the bevel.

If it is one, then each face is reduced in size and each edge becomes a single new face. *Tri* and *Quad* faces are created as necessary at the corresponding vertices. If the Recursion number is greater than one, then the bevel procedure is applied that number of times. Hence, for a Recursion of **2** each edge is transformed into 4 edges, three new faces appear at the edge while smoothing the original edge. In general the number of new edges is 2 elevated to the Recursion power.

Width

You can change the bevel width by moving the mouse towards and away from the object. The scaling can be controlled to a finer degree by either holding Ctrl, to scale in 0.1 steps, or by holding Shift to scale in 0.001 steps. LMB finalizes the operation, RMB or Esc aborts the action.

Alternatively, you can manually enter a scaling value by pressing Space. A popup dialog appears, , asking you to type in the beveling scale factor labeled as "Width". The scale is limited to a range from **0.0** to **10.0** and upon hitting "OK" the bevel action is completed.

Bevel window header.

Hints

Remember that in each recursion, for each new edge two new vertices are created, with additional vertices created at the intersection between edges. This means your vertex count can quickly become enormous if you bevel with a high recursion!

A Bevel, applied to a Curve object, forms a skin for the curve, like the outside of a cord, or hose pipe. Normally the Bevel is round like a pipe or soda can, but it can be rectangular for simulating wrought iron, oval with a crease for a power cord, star−shaped for a shooting star illustration; anything that can be physically formed by extrusion (extruded).

A Taper, applied to a Beveled Curve, changes the diameter of the Bevel along the length of the curve, like a python just having eaten a rat, or like a hose bulging up under pressure, or a vine growing.

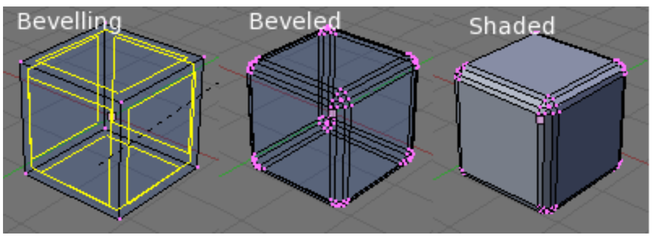
Bevel Selected Edges

With the W−>Bevel command, all edges of a given mesh are bevelled. To bevel only selected edges, use the Bevel Center script instead.

Examples

(*Bevelling a cube*) is an example of bevelling a cube with a Recursion of **2**. Once the Recursion number is set each face of the mesh receives a yellow highlight. The cube labeled **Bevelling** is the tool in action.

The final result can be seen in the grid labeled "Beveled" or "Shaded".



Bevelling a cube

Previous: Manual/Advanced Mesh Tools Contents *Next:* Manual/Vertex Groups

Snap to Mesh

Mode: Edit mode/Object mode

Hotkey: Shift Tab

Menu: Mesh/Object â†’ Transform â†’ Snap

Description

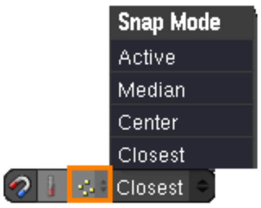
Used in conjunction with Grab, Extrusion, Scale or Rotation modes. Once the tool is activated you are ready to drag the surface to its destination.

Move your mouse pointer to where you want to snap and hold down Ctrl, move your pointer to adjust.

When satisfied with the result, confirm with LMB

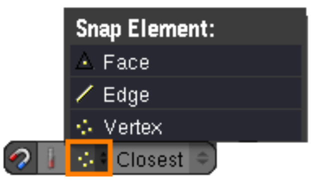
Options

Snap Mode



- Closest: **move the closest point of the selection to the target**
- Center: **move the current transformation center to the target. Can be used with 3D cursor to snap with an offset.**
- Median: **move the median of the selection to the target.**
- Active: **move the active element (vertex in edit mode, object in object mode) to the target.**

Snap Element



- Vertex
- Edge
- Face

The following shows different Element and Mode options. [Download Demo Video](theora)

Align Rotation

Align Object's Z axis with the normal of the target element.

Normals are interpolated between the two vertex normals when snapping to edges, otherwise, face normals and vertex normals are directly used.

Only works with Translations (Grab) done in object mode.

The following video shows a tree being snapped and aligned on a ground mesh. [Download Demo Video](theora)

User Manual: Contents | Guidelines | Blender Version **2.42**

A mesh is a set of connected Vertices, sometimes thousands of vertices for the more complex objects. Blender allows you to group these vertices for several main reasons:

- Re−using parts of a mesh for making copies
- Hiding "everything else" while you work on details
- Documentation and explanation to others
- Armatures deformation
- Generating particles from only the group
- Controlling the velocity of particles emitted

Armatures

Vertex Groups can be automatically created for each bone in an armature. However, that process is pretty involved and for more information on Armatures and Bone Vertex Groups, click here. The rest of this section will focus on user−defined vertex groups.

Why use Vertex Groups?

You want to re−use part of your object if that object has or maybe will have many of those parts. For example, a cabinet has many hinges and may have many knobs and doors; a chair or table has four legs; a fence has many posts. While you could model each of these parts independently as separate objects and parent them all together, sometimes you may wish to simply think of them as parts of an integral whole. While similar to each other, you may wish to alter each one slightly. For example, put a nick in one of the chair legs, or make some knobs larger or more ornate that others.

Use Vertex Groups to identify sub−parts of your model so that you can easily select and work on only that part. Especially using the hide function, vertex groups make it very easy to select a part of your model and hide everything else so that you can concentrate on only that part.

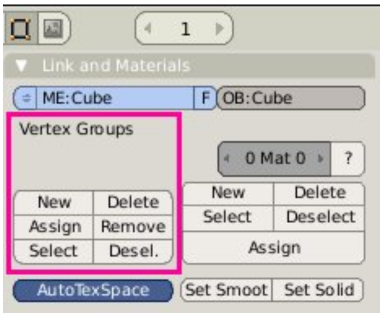
Vertex groups also make it easy to cull out and duplicate a part of the mesh many times. Consider modeling a Legoâ„¢ block. The most simplest block consists of a base and a nipple. To create a four−nipple block, you would want to be able to easily select the nipple vertices, and, still in edit mode, duplicate them and position them where you want them.

Another use for vertex groups is for skinning an armature. If you want to animate your mesh and make it move, you will define an armature which consists of a bunch of invisible bones. As a bone moves, it deforms or moves the vertices associated with it. Not all of the vertices, but some of them; the ones assigned to it. So, when you move the bone "Arm", the arm bone moves the Arm vertices, and not the Leg vertices. In this way, parts of the mesh can stretch and move around, while other parts remain stationary.

By entering the name of the group in the VGroup: field in the Particle and/or Particle Motion panels, the weight painting of the group will define how much particles come out. Recall that hair is a static particle; so define a Vertex Group called "Scalp" and use it to tell Blender to emit hair from the Scalp.

Another great use for vertex groups is for keeping track of vertex selections. For instance, I was modeling a coin−like object with a raised, beveled border. For some reason, I couldn't do a loop select around the base of the border, so I had to carefully select the vertices. I knew I'd need to use that same selection several times in the next few minutes while working on the model, so I named and saved the group, saving me a lot of work later.

Creating a Vertex Group



Vertex Group default panel

By default, an object does not have any groups, and all of its vertices are hanging out there in cyberspace as loners. The image to the right highlights the Vertex Groups buttons in hot pink. These buttons are located in a Buttons window in the Editing F9 buttons group on the Link and Materials panel. They are shown when an object with vertices is selected AND being edited (Tab). You can tell when an object is in Edit mode because your 3D window cursor is a cross−hair.

Only Groups are for Vertices

Vertex Groups are only available for objects that have vertices. Text objects, for example, cannot have vertex groups and the panel is not shown when that kind of object is selected. Vertex Groups are only shown when an object with vertices is being edited.

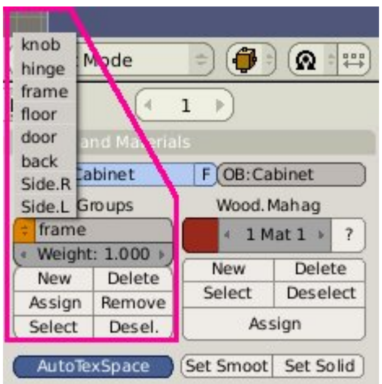
To create a vertex group, LMB click the New button. When you do, a new vertex group (named, surprisingly, "Group") is created, and the panel shows you a Weight numeric slider/entry/scroll box. Any selected vertices are not yet assigned to the new vertex group, you must click the Assign button to actually allocate vertices to the newly created vertex group.

Check Your Assignment

It's a good idea to make sure the vertices have been properly assigned to the group by using the deselect and select buttons. If nothing happens, just hit the Assign button to add the selected vertices to the group.

Naming Vertex Groups

To name a group something other than the creative "Group", Shift LMB click the name field, and type in the name you want.



Cabinet Vertex Group example

For example, consider the model of a kitchen cabinet. The cabinet consists of three vertical walls (two sides and a back), a floor and countertop, a door frame, a door, a knob and two hinges. You may or may not, at some point, to be able to model the door opening. You may want to make the cabinet a single door or later easily modify it to be a double door (with two knobs). You may wish to copy the knob design and use it for the drawers which you will be modeling later. In this case, you would want to define at least three vertex groups: Base, Door, and Knob. If you were writing a user manual, you would want your example to contain each possible group for maximum re−use and selection, as shown.

Access the group list by clicking the list selector button next to the group name. Select a group by clicking on any named group.

Assigning Vertices to a Group

To add vertices to a group you do the following:

1. Select the group you want to work with from the group list.
2. Use your mouse to Shift RMB select more vertices that you want in that group.
3. LMB click the Assign button

Keep in mind that a vertex can be assigned to multiple groups.

Note

When using the Assign button to assign selected vertices to a vertex group, any vertices that were already in the vertex group are not removed, so Assign adds extra vertices to the selected vertex group.

Seeing a Vertex Group

From experience, we have found that is is best to start first by seeing the existing vertices in a group, before adding more or removing some. To do this, de−select all vertices by pressing A once or twice in the 3d window until the User Preferences header shows **Ve:0−**x, where x is the number of vertices in your mesh. This means that zero (0) vertices are selected. The Vertex count is located just to the right of the Blender Version.

Then, with the appropriate group active, press the Select button. In your 3D window, the vertices that belong to the active group will be highlighted.

Removing Vertices from a Group

To remove vertices from a group:

1. Select the vertices you want to remove from the vertex group.
2. Select the group you want to work with from the group list.
3. LMB click the Remove button.

Deselecting Vertices

Sometimes you will want to see if any vertices are still loners. To do so, select All the vertices in the 3D window. For each Vertex Group, LMB click the Desel. button to de−select the vertices in that group. Repeat the de−selection for each group. When you run out of groups, any vertices left highlighted are the loners. Sort of like picking baseball teams.

Deleting a Group

To delete a vertex group, select it from the list and click Delete. Yes, it's as simple as that. Any vertices that belonged to that group are unassigned from that group. However, please keep in mind that vertices can belong to many Groups. When they are unassigned from one group, they still belong to their other groups.

Using Vertex Groups in Practice

Assume you have defined the groups used in our cabinet example. Here are some examples of common things you might want to do involving Vertex Groups.

Duplicating Parts

You now want to make that cabinet a double door model:

1. Select the cabinet object (RMB) and enter Editmode (Tab).
2. Ensure that NO vertices are selected (Ve:0 − remember?).
3. Select the "Knob" vertex group from the dropdown menu.
4. Click the Select button.
5. Move your mouse into the 3D window.
6. Duplicate that sub−mesh by pressing Shift D. The vertices are copied, selected, and grabbed.
7. Move the mouse over to position the new knob.
8. LMB to drop the sub−mesh.

The duplicated vertices belong to the same group(s) as their originals.

To assign this new knob to its own group, click New, name it something like "Knob.L" and click on Assign. See #Creating a Vertex Group and #Assigning Vertices to a Group.

Left and Right naming convention

Certain features of Blender can perform related actions on groups that are left and right counterparts of each other. If you end a name in ".L" or ".left" and its counterpart ".R" or ".right", Blender may be able to easily mirror its actions for you. You can read more about the naming convention in Manual/Editing Armatures: Naming conventions. The convention for armatures/bone apply here as well.

Simplifying a Vertex Group

You may have correctly surmised that the original Knob group now has both sets of vertices: the original and the duplicated ones. You've created a "Knob.L" group, but there is no corresponding 'right' group. The Knob group really needs to be renamed and contain only the vertices for the right knob. To correct this,

1. Ensure the new "Knob.L" vertices still selected (the ones that don't belong)
2. Select the original Knob Vertex Group from the list
3. Click the Remove button

To test your work, deselect all vertices and click the Select button. Only the vertices from the original knob should highlight. Rename this group "Knob.R"

Repeat the above for the "door" and "hinge" group, and you now have a two−door cabinet model. Note that you will have to either make the doors narrower or cabinet wider to accommodate the new door.

Combining Groups

To create a Knobs (plural group), you could:

1. Ensure that no vertices are selected.
2. Select the "Knob.L" group (select its name from the list and click Select)
3. Select the "Knob.R" group (ditto).
4. Observe that selecting one set of vertices does not deselect the others; the selection process adds on vertices to the selection.
5. Click the New button, and name the group "Knobs"

Focus on a part of your model

You want to make an inset panel on the door. To work on the door sub−mesh without cluttering up your screen with all the other vertices, you would:

1. Ensure that ALL vertices are selected. (You can use A for that.)
2. Deselect the "door" group by selecting its name from the Vertex Group list and clicking Desel. (for DeSelect), leaving everything BUT the door selected.
3. With your cursor in the 3d Window, Hide the selected vertices. Poof! They disappear.

Separating a part into its own

Now, the patent lawyer calls and says that you must patent your hinge design to keep anyone else from copying it; you need to separate the hinge out from the cabinet mesh:

1. Ensure that NO vertices are selected.
2. Select the Hinge vertices (select the name from the Vertex Group list, and click Select)
3. With your cursor in the 3d Window, seParate them into their own object.
4. The remaining cabinet vertices are left. Tab out of editmode and RMB click the floating hinge object. Note that it is conveniently called "Cabinet.001", and has all the same Vertex Groups as the original. Delete those groups you do not need, rename the object "Hinge".

Parent it to the original (and now hinge−less) "Cabinet" object (include the parent by Shift RMB clicking the Cabinet, and pressing Ctrl P). Now, when you move your cabinet, the hinges move with it.

Finding Groups for a Vertex

As you are rigging and animating the deformation of a mesh, you might need to find out which groups a vertex belongs to, and to adjust the weights of influence each group has on that particular vertex.

1. Select the vertex
2. In Edit mode, press N to open the transform properties window for that vertex.
3. Open the drop−down menu that shows all vertex groups to which it belongs.
4. From this window you also can assign weights to the vertex for each group.

About Weight

By default, every vertex in a group has a weight of 1.00. If a vertex belongs to multiple groups, it has a combined weight. When influenced by a bone or other object, it is moved by an amount proportional to its weight; heavier vertices move less. So, a middle vertex belonging to two groups (each with a weight of 1.00) would move half as much as a left vertex that only belonged to one group. This weighting system provides realistic deformation of a mesh when bones move, for example, around the shoulder area, where some of the vertices belong to both the chest and the arm groups.

You can set the weight of all vertices in a group using the Weight numeric control. For more advanced weighting, please read Weight Painting. Weight Painting allows you to smoothly blend individual vertex weights so that meshes deform smoothly.

Previous: Manual/Edge and Face Tools	Contents	*Next:* Manual/Weight Paint

User Manual: Contents | Guidelines | Blender Version **pre2.43**

The Weight Paint Mode is used to create and modify Vertex Groups. A vertex may not only be a member of one or more Vertex Groups, but also may have a certain weight in each group. The weight symbolises its influence on the result.

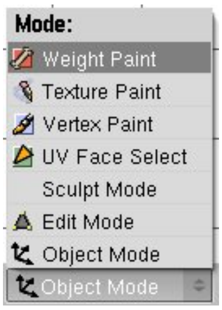


Image 1: Changing to Weight Paint Mode.

Mode: Object Mode

Hotkey: Ctrl Tab

Menu: Mode Menu (**Image 1**)

When you change to Weight Paint Mode you see the selected object (if you have not created any groups yet), in a slightly shaded, blue color (**Image 2**).

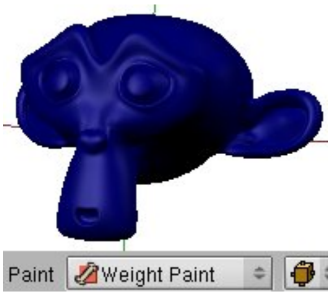


Image 2: An object in Weight Paint−Mode.

The color visualises the weight of each vertex of the currently active group. A vertex drawn in blue indicates either : a weight of zero, not in the active group, or not in any group at all.

You assign the weight of each vertex by painting on the mesh with a certain color. Starting to paint on a Mesh will automatically create a new Vertex Weight Group (when no group existed or is active). If a vertex doesn't belong to the active group it is automatically added (if the option Vgroup is not set), even if you paint with a weight of "0". The used color spectrum is shown in **Image 3**.

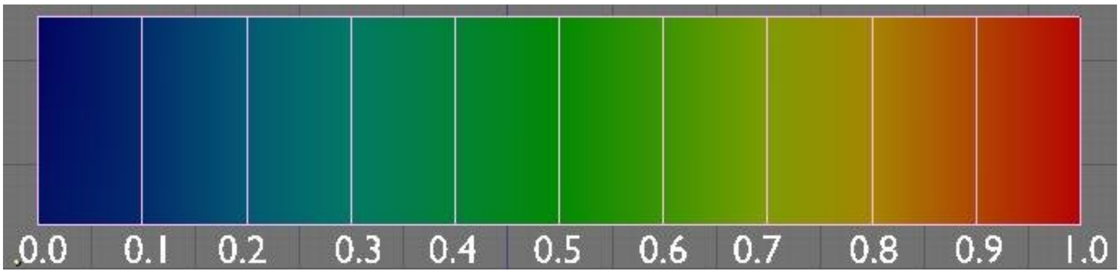


Image 3: The color spectrum and their respective weights.

You paint on the mesh with a brush. The color only influences the vertices, neither the faces nor the edges. So don't try and paint these. There is a tool panel for the brush in the Editing−Buttons (F9) as well as in the 3D Window (press N to open it).

Weight−Painting survival tips:

A few tips that will save you some hassle when painting weight:

- Press F in Weight Paint Mode to activate the *Face Select* mode. Now you can select faces to paint and Hide faces like in Edit Mode.
- Press B to Border−select faces to paint using LMB. Use the RMB border select to exclude the selected faces from painting.
- Draw a *Clipping Border* with Alt B. It will separate a visible part of the 3D−window. You can draw only in this part. If you press Alt B again the *Clipping Border* will be removed.
- Turn off Soft in the Paint−Panel. If you have Soft activated you will reach the target value only after several repeated paint actions, and it's especially difficult to reach "0.000".

Paint−Panel

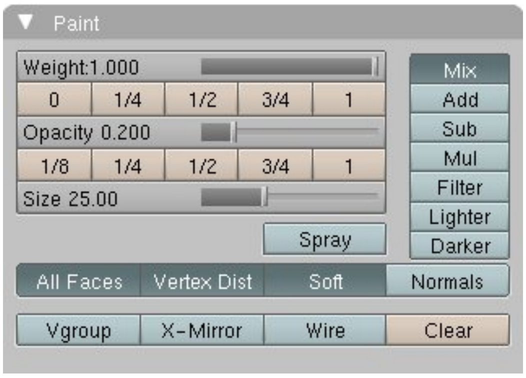


Image 4: The Paint−Panel in the Editing−Buttons.

The tools in the Paint panel are sophisticated, and you can apply weight in the finest nuances. But normally you won't need all these options, and you will apply weight using a few techniques. The most used and important settings are **drawn in bold**.

- **Weight:** The weight (color) that is used to paint. The button row below contains five weight presets to paint. By default, painting works with a fixed amount relative to the previous state (like Gimp or Photoshop defaults), so you can add for example "0.2 weight" to vertices while keeping the mouse button pressed.
- **Opacity:** The extent to which the vertex colour changes while you are painting.
- Size: *The size of the brush, which is drawn as a circle during painting.*
- Spray: The Spray option accumulates weights on every mouse move.
- Mix/Add/...: The manner in which the new color replaces the old when painting.
    - Mix: The new color replaces the old color. Applying more and more of the new color will eventually turn it the new color.
    - Add: The new color is added to the old. For example, painting green over a red area will make it yellow, since green and red make yellow in the RGB world.

    - ♦ Sub: The new color is subtracted from existing. For example, painting with green over a yellow area will make it red.
    - ♦ Mul: The new color is multiplied by the old.
    - ♦ Filter: Paint based on alpha value.
    - ♦ Lighter: Decreases the saturation (amount) of that color
    - ♦ Darker: Increases the saturation of the color
- All faces: If this is turned off, you will only paint vertices which belong to a face on which the cursor is. This is useful if you have a complicated mesh and you would paint on visually near faces that are actually quite distant in the mesh.
- Vertex Dist: Paints only on vertices which are actually under the brush. If you switch this off, all vertices belonging to faces touched by the brush are painted. If you have a sparse mesh and use subsurfaces you want to keep this on.
- **Soft:** This specifies that the extent to which the vertices lie within the brush also determine the brush's effect. It's extremely difficult to paint with zero then. You want to turn this off in all normal situations.
- Normals: The vertex normal (helps) determine the extent of painting. This causes an effect as if painting with light.
- **Vgroup:** Only vertices which belong to the active vertex group are painted. Very useful for clearing up and refining vertex groups without messing other groups up.
- **X−mirror:** Use the X−Mirror option for mirrored painting on groups that have symmetrical names, like with extension .R .L or _R or _L. If a group has no mirrored counterpart, it will paint symmetrical on the active group itself. You can read more about the naming convention in [Manual/Editing Armatures: Naming conventions]. The convention for armatures/bone apply here as well.
- Wire: Show additionally the wireframe of the objects. Since it shows the subsurfaced wire it's quite useless. It's better to use the *Select*−Mode (see below).
- **Clear:** Removes all vertices from the active group.

Tools

If you have a complex mesh it is nearly impossible to reach all vertices in Weight Paint mode. And it is quite difficult to tell where the vertices are exactly. But there's a very good and easy solution: the *Select* Mode.

Mode: Weight Paint Mode

Hotkey: F

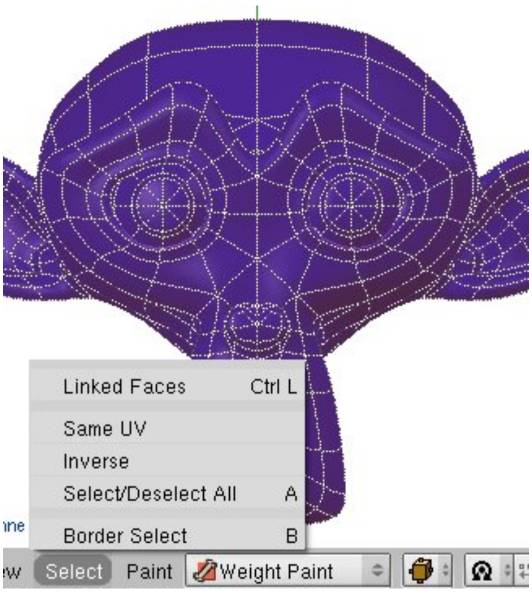


Image 5: Select−Menu in Weight Paint−Mode.

Select Mode has many advantages over the normal Weight Paint mode.

1. The original mesh is drawn, even when subsurface is active. You can see the vertices you have to paint over. You need to activate Vertex Colors for the mesh (Editing Buttons−>Mesh Panel−>VertCol−>Make) to see the Weights in *Select* mode.
2. You can select faces, only the vertices of selected faces are painted on.
3. Selecting tools include:
    - RMB − Single faces. Use Shift RMB to select multiple.
    - A − All faces, also to deselect.
    - Alt B − Block/Box selection.
    - B−B − Select with brush.
    - Ctrl L − Select linked.
    - In the Select−Menu: Faces with Same UV, also invert selection (Inverse).
4. You may hide selected faces with H and show them again with Alt H (**Image 6**).

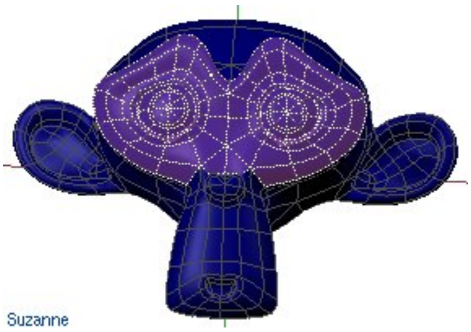


Image 6a: Select interfering faces ...

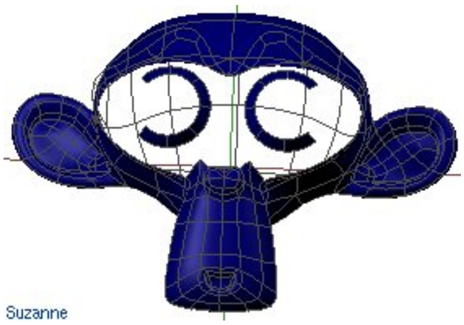


Image 6b: ... and hide them with H.

Image 7: A vertex belonging to two vertex groups.

To constraint the paint area further you may use the *Clipping Border*.

- Press Alt B and LMB −Drag a rectangular Area. The rest of the 3D window is hidden.
- If you want to know which groups a Vertex belongs use Shift LMB . You can change between the groups in the appearing Pop−Up Menu (**Image 7**).
- N in the 3D−window opens a Paint−panel instead of the transform properties panel (**Image 8**).

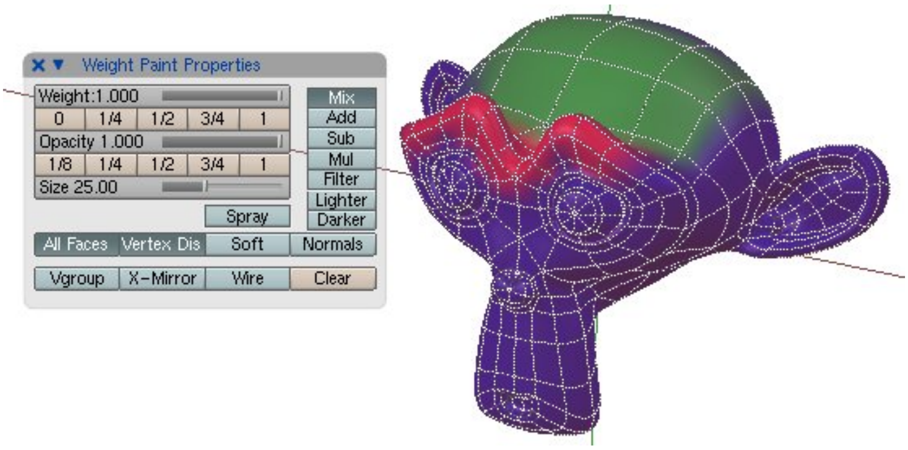


Image 8: *Weight Paint Properties* Panel in the 3D−Window.

Scripts in *Paint*−Menu

Weight Gradient:
: This script is used to fill the selected faces with a gradient (**Image 3 & 9**). To use the script, select the faces you wish to apply the gradient to. Click twice on the mesh to set the start and end points of the gradient. The color under the mouse will be used for the start and end blend colors, so you have to weight paint first. Holding Shift or clicking outside the mesh on the second click will blend the first colour to nothing.

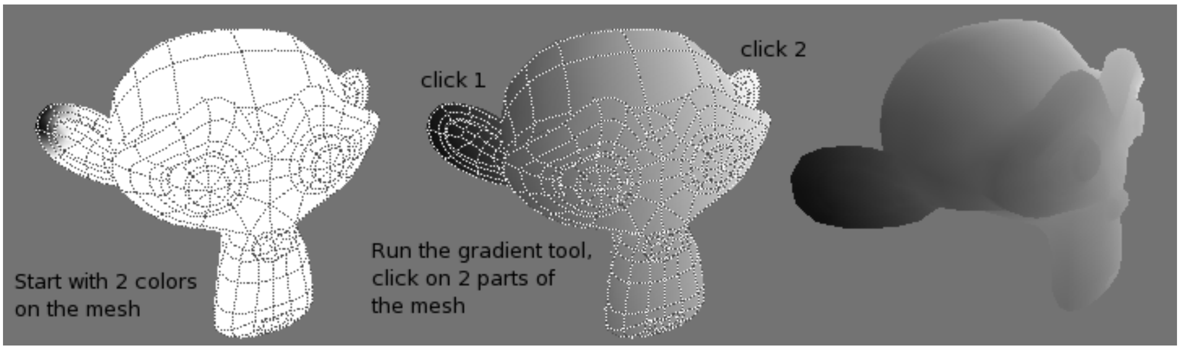


Image 9: Setting the gradient and result.

Normalise/Scale Weight:
: Maximizes weights to a user set peak and optionally scales other groups too to keep the proportion of the weights even.

Grow/Shrink Weight:
: Uses the mesh topology to expand/contract the vert weights (works like colorbleeding).

Clean Weight:
: Removes vertices with low weights from the current group.

Weight−Painting for Bones

This is probably the most often used application of weight painting. When a bone moves, vertices around the joint should move as well, but just a little, to mimic the stretching of the skin around the joint. Use a 'light' weight (10−40%) paint on the vertices around the joint so that they move a little when the bone rotates. While there are ways to automatically assign weights to an armature (see the Armature section), you can do this

manually. To do this from scratch, refer to the process below. To modify automatically assigned weights, jump into the middle of the process where noted:

- Create an Armature
- Create a Mesh that will be deformed when the armature's bone(s) move
- With the Mesh selected, create an Armature modifier for your mesh (located in the editing buttons, modifier panel). Enter the name of the Armature

(Pick up here for modifying automatically assigned weights)

- Select the Armature in 3D View, and bring the armature to **Pose mode** (in the 3D View window header mode selector).

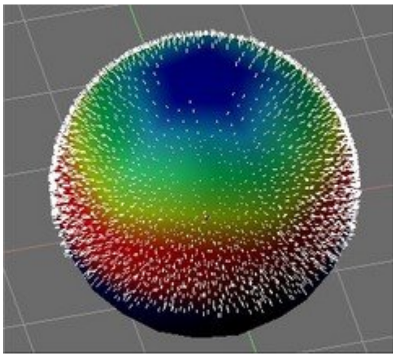
- Select a desired bone in the Armature
- Select your mesh (using RMB and change immediately to Weight Paint mode. The mesh will be colored according to the weight (degree) that the selected bone movement affects the mesh. Initially, it will be all blue (no effect).
- Weight paint to your heart's content. The mesh around the bone itself should be red (generally) and fade out through the rainbow to blue for vertices farther away from the bone.

You may select a different bone with RMB . If the mesh skins the bones, you will not be able to see the bones because the mesh is painted. If so, turn on X−Ray view (Buttons window, Editing buttons, Armature panel). While there on that panel, you can also change how the bones are displayed (Octahedron, Stick, B−Bone, or Envelope) and enable Draw Names to ensure the name of the selected bone matches up to the vertex group.

If you paint on the mesh, a vertex group is created for the bone. If you paint on vertices outside the group, the painted vertices are automatically added to the vertex group.

If you have a symmetrical mesh and a symmetrical armature you can use the option X−Mirror. Then the mirrored groups with the mirrored weights are automatically created.

Weight−Painting for Particles



Faces or vertices with zero weight generate no particles. A weight of 0.1 will result in 10% of the amounts of particles. This option "conserves" the total indicated number of particles, adjusting the distributions so that the proper weights are achieved while using the actual number of particles called for. Use this to make portions of your mesh hairier (is that really a word?) than others by weight−painting a vertex group, and then calling out the name of the vertex group in the Object Particles panel, in the VGroup: field.

Previous: Manual/Vertex Groups	Contents	*Next:* Manual/Subsurf Modifier

User Manual: Contents | Guidelines | Blender Version **2.45**

Subdivision Surfaces

Mode: Any Mode

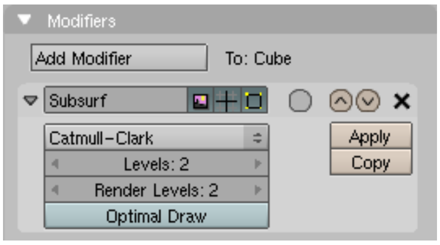
Panel: Editing Context Modifiers

Hotkey: F9 (Panel) Shift O (Toggle SubSurf in Object Mode)

Description

A Subdivision Surface is a method of subdividing the faces of a mesh to give a smooth appearance, to enable modelling of complex smooth surfaces with simple, low−vertex meshes. This allows high resolution Mesh modelling without the need to save and maintain huge amounts of data and gives a smooth *organic* look to the object. With any regular Mesh as a starting point, Blender can calculate a smooth subdivision on the fly, while modelling or while rendering, using Catmull−Clark Subdivision Surfaces or, in short SubSurf.

Options



Modifiers panel.

SubSurf is a Modifier. To add it to a mesh, press Add Modifier and select Subsurf from the list.

- Levels defines the display resolution, or level of subdivision.
- Render Levels is the levels used during rendering. This allows you to keep a fast and lightweight approximation of your model when interacting with it in 3D, but use a higher quality version when rendering.
- To view and edit the results of the subdivision ("isolines") on the Editing Cage while you're editing the mesh, click in the gray circle next to the arrows for moving the modifier up and down the stack. This lets you grab the points as they lie in their new subdivided locations, rather than on the original mesh.
- Optimal Draw restricts the wireframe display to only show the original mesh cage edges, rather than the subdivided result to help visualisation.

Hints

You can use Shift O if you are in ObjectMode to switch Subsurf On or Off. To turn the subsurf view off (to reduce lag), press Alt+Shift+O. The Subsurf level can also be controlled via Ctrl 1 to Ctrl 4, but this only affects the visualization sub−division level.

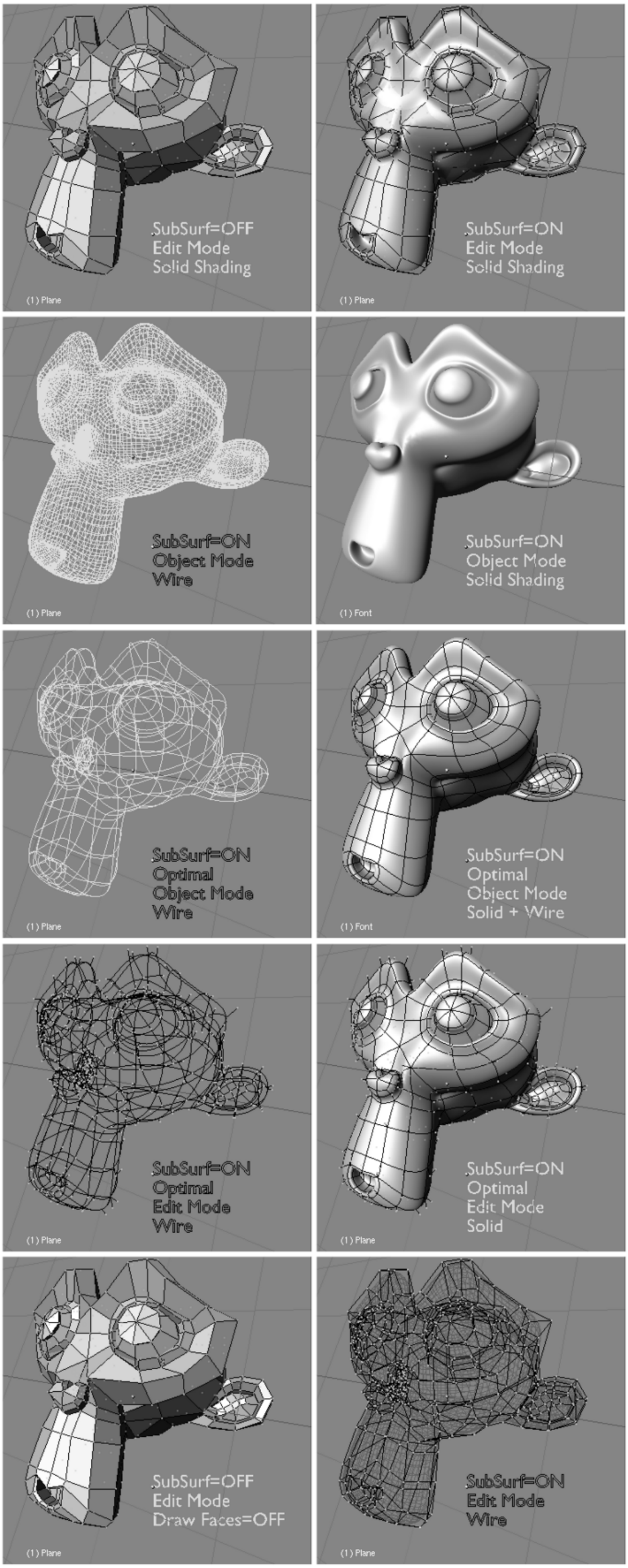
A SubSurfed Mesh and a NURBS surface have many points in common such as they both rely on a "coarse" low−poly "mesh" to define a smooth "high definition" surface, however, there are notable differences:

- NURBS allow for finer control on the surface, since you can set "weights" independently on each control point of the control mesh. On a SubSurfed mesh you cannot act on weights.
- SubSurfs have a more flexible modelling approach. Since a SubSurf is a mathematical operation occurring on a mesh, you can use all the modelling techniques described in this chapter on the mesh. There are more techniques, which are far more flexible, than those available for NURBS control polygons.

Since Subsurf computations are performed both real−time, while you model, and at render time, and they are CPU intensive, it is usually good practice to keep the SubSurf level low (but non−zero) while modelling; higher while rendering.

Examples

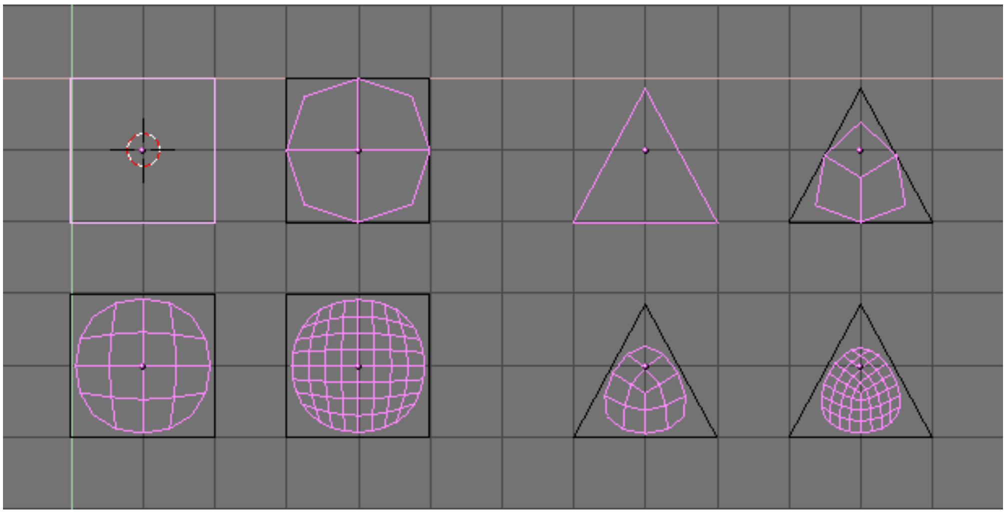
SubSurfed Suzanne shows a series of pictures showing various different combinations of Subsurf options on a Suzanne Mesh.


SubSurf=OFF
Edit Mode
Solid Shading
(1) Plane
SubSurf=ON
Edit Mode
Solid Shading
(1) Plane
SubSurf=ON
Object Mode
Wire
(1) Plane
SubSurf=ON
Object Mode
Solid Shading
(1) Font
SubSurf=ON
Optimal
Object Mode
Wire
(1) Plane
SubSurf=ON
Optimal
Object Mode
Solid + Wire
(1) Font
SubSurf=ON
Optimal
Edit Mode
Wire
(1) Plane
SubSurf=ON
Optimal
Edit Mode
Solid
(1) Plane
SubSurf=OFF
Edit Mode
Draw Faces=OFF
(1) Plane
SubSurf=ON
Edit Mode
Wire
(1) Plane

SubSurfed Suzanne.

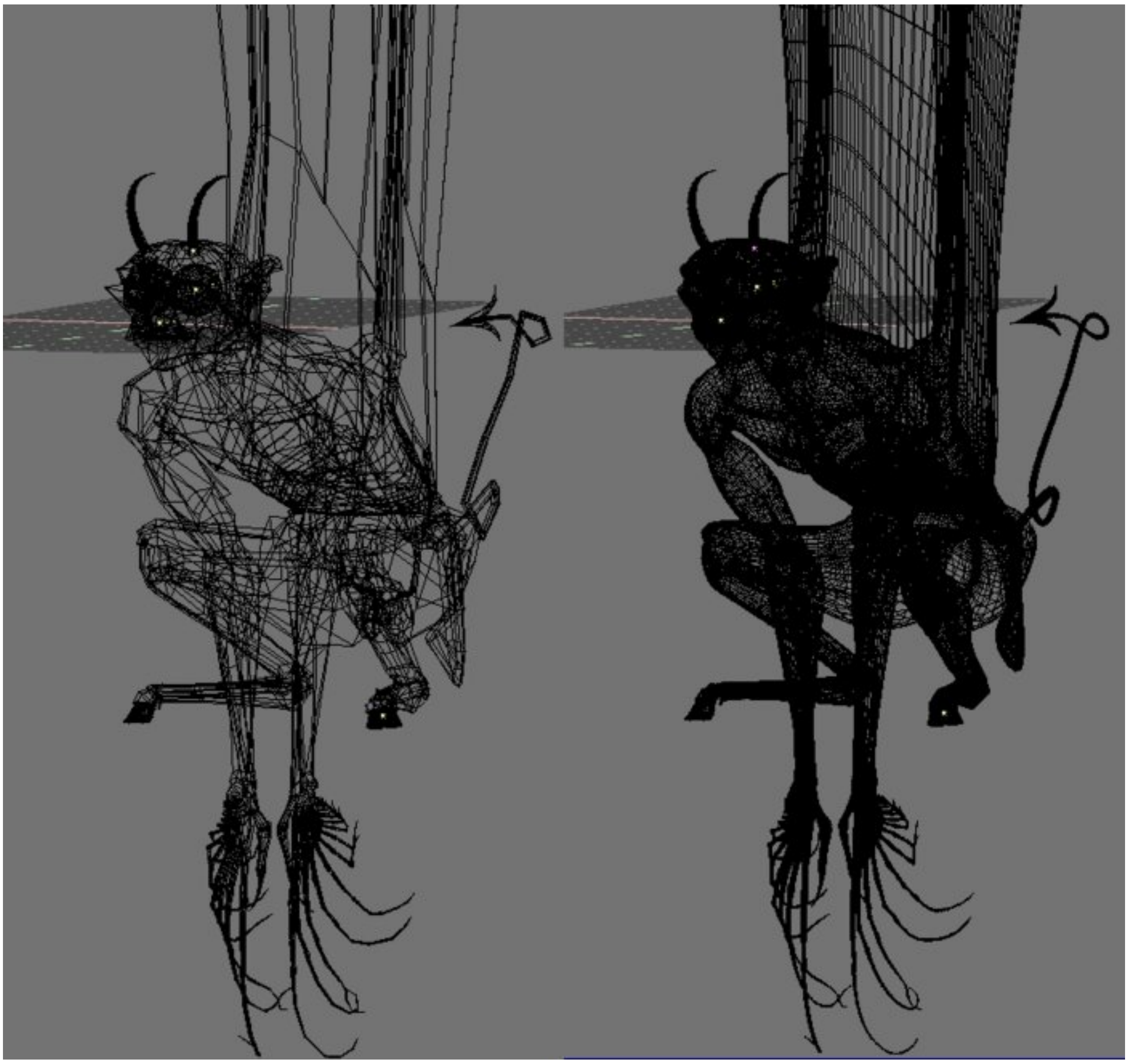
SubSurf of simple square and triangular faces. shows a 0,1,2,3 level of SubSurf on a single square face or on a single triangular face. Such a subdivision is performed, on a generic mesh, for *each* square or triangular face.

It is evident how each single quadrilateral face produces 4^n faces in the SubSurfed mesh. *n* is the SubSurf level, or resolution, while each triangular face produces $34^{(n-1)}$ new faces (*SubSurf of simple square and triangular faces.*). This dramatic increase of face (and vertex) number results in a slow−down of all editing, and rendering, actions and calls for lower SubSurf level in the editing process than in the rendering one.

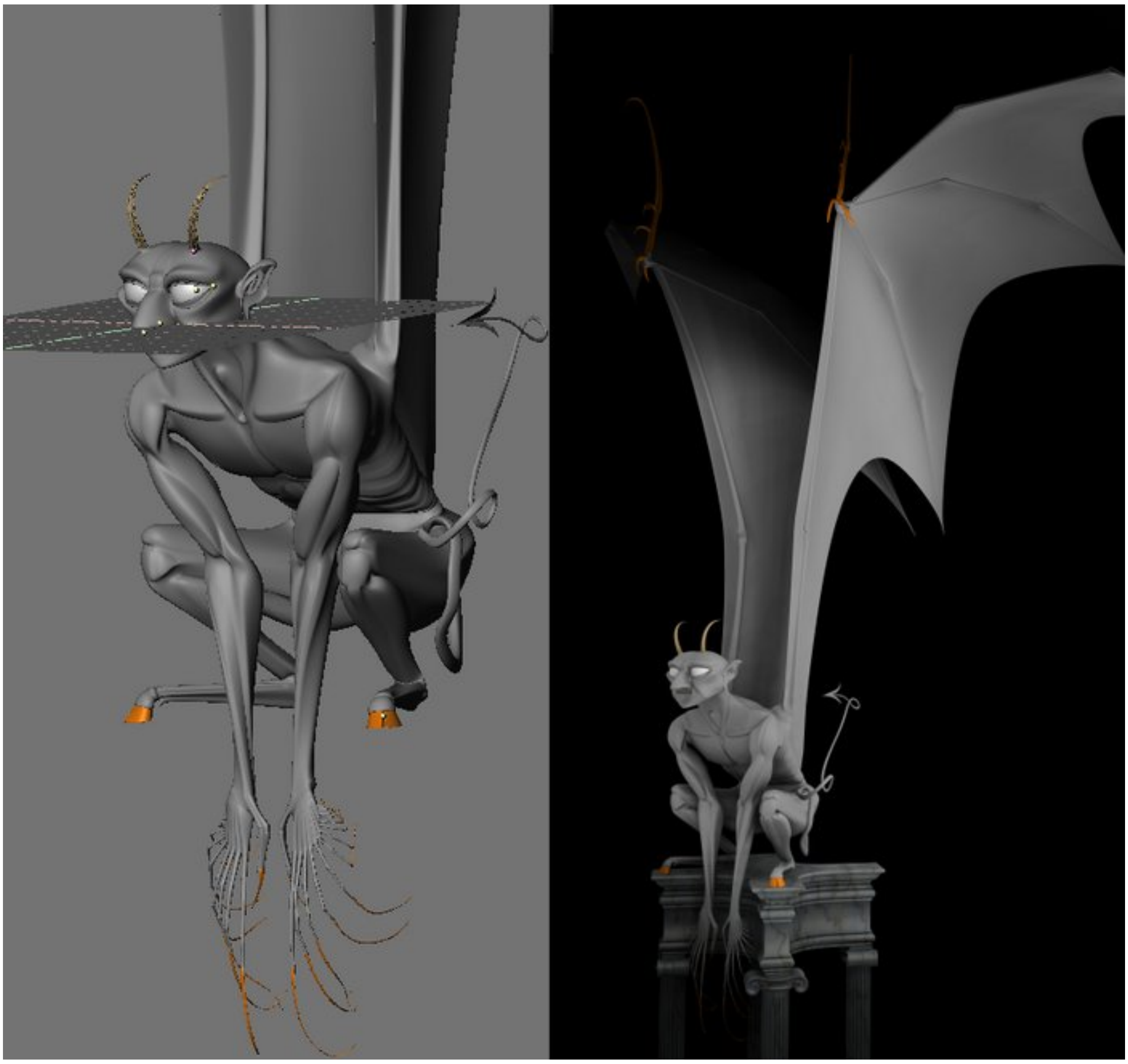


SubSurf of simple square and triangular faces.

The SubSurf tool allows you to create very good "organic" models, but remember that a regular Mesh with square faces, rather than triangular ones, gives the best results. *A Gargoyle base mesh (left) and pertinent level 2 SubSurfed Mesh (right).* and *Solid view (left) and final rendering (right) of the Gargoyle.* show an example of what can be done with Blender SubSurfs.



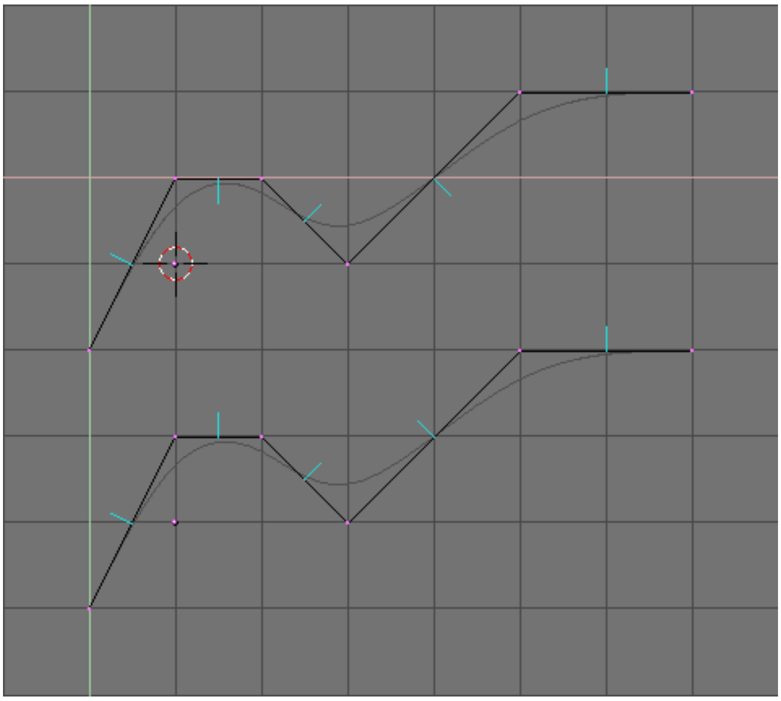
A Gargoyle base mesh (left) and pertinent level 2 SubSurfed Mesh (right).



Solid view (left) and final rendering (right) of the Gargoyle.

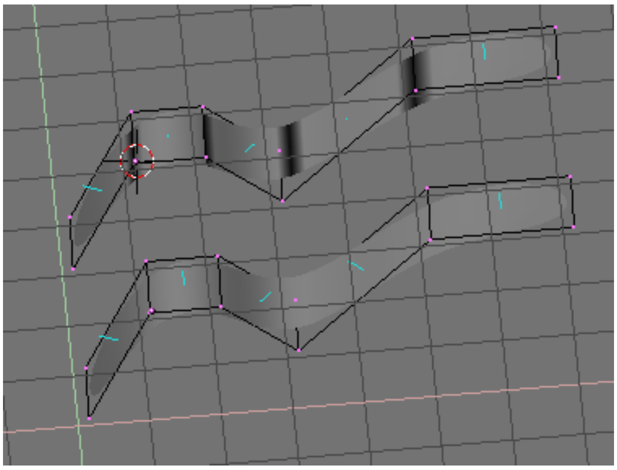
Limitations & Work−arounds

Blender's subdivision system is based on the Catmull−Clark algorithm. This produces nice smooth SubSurf meshes but any 'SubSurfed' face, that is, any small face created by the algorithm from a single face of the original mesh, shares the normal orientation of that original face.



Side view of subsurfed meshes. With random normals (top) and with coherent normals (bottom)

This is not an issue for the shape itself, as *Side view of subsurfed meshes. With random normals (top) and with coherent normals (bottom)* shows, but it is an issue in the rendering phase and in solid mode, where abrupt normal changes can produce ugly black lines (*Solid view of SubSurfed meshes with inconsistent normals (top) and consistent normals (bottom).*).

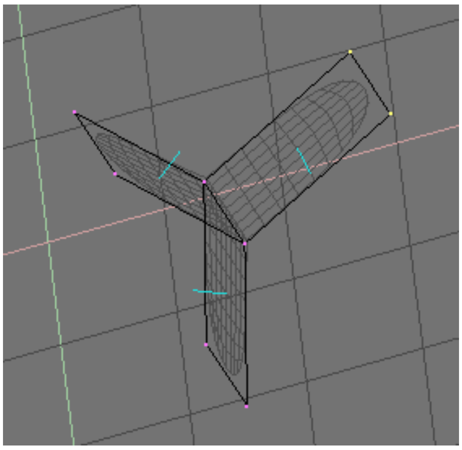


Solid view of SubSurfed meshes with inconsistent normals (top) and consistent normals (bottom).

Use the Ctrl N command in EditMode, with all vertices selected, to recalculate the normals to point outside.

In these images the face normals are drawn cyan. You can enable drawing normals in the EditButtons (F9) menu.

Note that Blender cannot recalculate normals correcty if the mesh is not "Manifold". A "Non−Manifold" mesh is a mesh for which an 'out' cannot unequivocally be computed. From the Blender point of view, it is a mesh where there are edges belonging to *more* than two faces.



A "Non−Manifold" mesh

A "Non−Manifold" mesh shows a very simple example of a "Non−Manifold" mesh. In general a "Non−Manifold" mesh occurs when you have internal faces and the like.

A "Non−Manifold" mesh is not a problem for conventional meshes, but can give rise to ugly artifacts in SubSurfed meshes. Also, it does not allow decimation, so it is better to avoid them as much as possible.

Use these two hints to tell whether a mesh is "Non Manifold":

- The Recalculation of normals leaves black lines somewhere
- The "Decimator" tool in the Mesh Panel refuses to work stating that the mesh is "Non Manifold"

Weighted creases for subdivision surfaces

Mode: Edit Mode (Mesh)

Panel: 3D View Transform Properties

Hotkey: Shift E or N (Transform Properties Panel)

Menu: Mesh Edges Crease Subsurf

Description

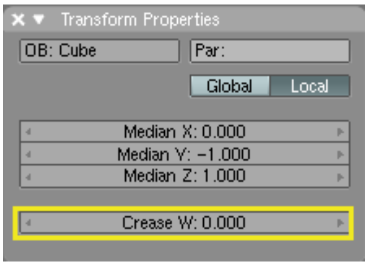
Weighted edge creases for subdivision surfaces allows you to change the way Subsurf subdivides the geometry to give the edges a smooth or sharp appearance.

Options

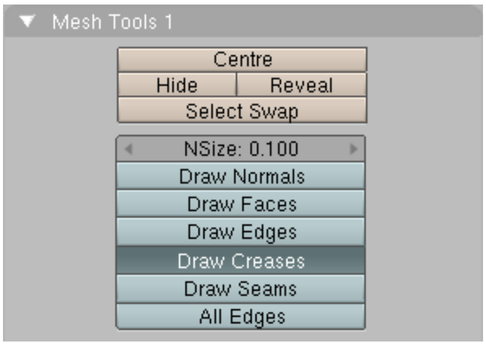
The crease weight of selected edges can be changed interactively by using Shift E and moving the mouse towards or away from the selection. Moving the mouse away from the edge increases the weight. You can also use Transform Properties (N) and enter the value directly. A higher value makes the edge "stronger" and more resistant to subsurf. Another way to remember it is that the weight refers to the edge's sharpness. Edges with a higher weight will be deformed less by subsurf. Recall that the subsurfed shape is a product of all

intersecting edges, so to make the edges of an area sharper, you have to increase the weight of all the surrounding edges.

You can enable an indication of your edge sharpness by enabling Draw Creases. See (*Mesh Tools 1 panel*).



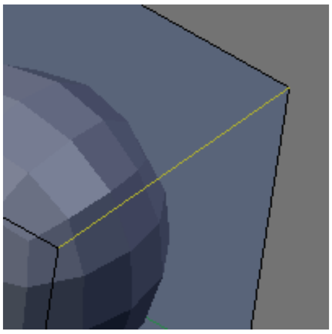
Transform Properties



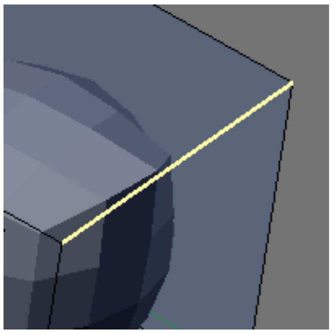
Mesh Tools 1 panel

Examples

The sharpness value on the edge is indicated as a variation of the brightness on the edge. If the edge has a sharpness value of 1.0, the edge will have a brighter color, and if sharpness value is 0.0, the edge will not be so bright.



Edge sharpness 0.0



Edge sharpness 1.0

Previous: Manual/Softbody Modifier	Contents	*Next:* Manual/UVProject Modifier

If you arrived at this page from either pages "Weight Paint" or "Multi Resolution Mesh" and wish to carry on navigating in the order laid out in the Mesh section of the Main Index page use the Navigation bar below:

Previous: Manual/Weight Paint	Contents	*Next:* Manual/Multires

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.43**

Multiresolution Mesh

Mode: Edit Mode, Object Mode

Panel: Editing Context Multires

Hotkey: F9 (Panel)

Description

Multires stands for 'multiple resolution mesh'. Multires allows you to edit the mesh at both high and low levels of complexity. Changes you make at one level of resolution propagate to all other levels.

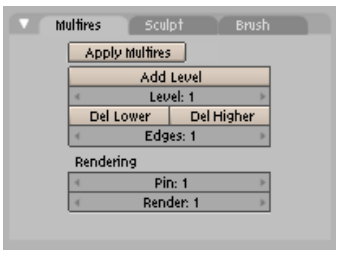
This feature is most often used for Sculpt Mode, where multires levels are added to a base mesh to sculpt with increasingly fine detail. While sculpting, viewing a high−complexity mesh is CPU−intensive. Multires allows the user to view the mesh at a lower level for positioning, lighting, and animating. Even when viewing a mesh at a low level, all the fine detail still remains in the multires data of the mesh and can be viewed at any time by setting the Level: value to a higher number.

Multires Monster head sculpt by Tom Musgrove

Options

In the multires panel press add multires. Then press add level to increase the levels of multires. Level 1 is the same as a mesh without multires. If you press delete multires it will convert the mesh to whatever the current level is that is selected. You can add multiple levels until you reach your ram limit.

Multires panel, no multires data added yet.



Multires panel.

- Add multires − Initializes the mesh to accept multires levels.
- Apply multires − Appears only after adding multires to a mesh. Removes all the multires data, *leaving the mesh at the currently set resolution*.
- Add Level − Changes to the highest level and subdivides the mesh. This also modifies the vertices in all lower levels to match those in the highest level.
- Del Lower − Applies the changes to the mesh at the current level and removes all lower mesh levels.
- Del Higher − Deletes all multires levels above the current level.
- Edges − Determines the maximum level of edges (the highlighted outline of a mesh) drawn. Edge display must be enabled in the Draw panel. Setting "Edges" to a high level will make the edges of the mesh closely follow the complexity of the mesh. This is especially useful for sculpting when you need to see the effect of your brush along the edge of a mesh.
- Pin − If you have a modifier on the mesh, this determines what level the modifier is applied during rendering. Any multires level above the modifier is disabled.
- Render − This determines what level of multires the model is rendered at. By default set to the highest available level.

Limitations

- Only the shape and not the topology of the mesh can be changed with multires enabled. Thus any tool that changes the topology (deleting or adding of faces) is deactivated.
- Multires currently incompatible with shapekeys.
- Some modifiers can result in a slow down of display and interaction if multires is also enabled

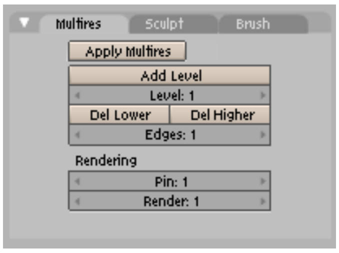
Workflow

Multres panel, no multires data added yet.

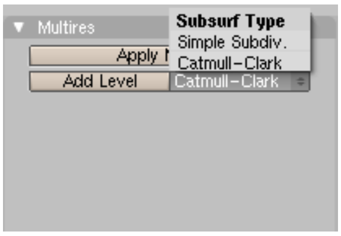
- Select a mesh. In the Multires panel of the Edit buttons, click the Add Multires button. This simply sets up the mesh for multiple resolutions. It does not add any levels.

Multires data added. Click Add Level to add the first level of resolution.

- Click Add Level to add the first level of multiresolution data.



Multires panel after clicking the "Add Level" button once. Click "Add Level" at this point to add higher complexity levels.



Simple subdivision or Catmull Clark? Decisions..

- With the first level of multires added, more options become available (see above). Level 1 is the same as the mesh without any extra resolution. To add levels with higher resolution than the original mesh, click Add Level again. You can choose between simple subdivision or Catmull Clark with the drop−down slider. Simple subdivisions are useful when you want your mesh to keep it's

existing shape without being smoothed.

- Adding more levels will increase the load on your CPU and make frame rate drop considerably. When you don't need to view the mesh in high detail, set the Level: value to Level 1 or 2.

- Enable Sculpt Mode and sculpt at the level that gives you a balance of performance and control.

See also

- BlenderDev/Multires − Development details and some examples.

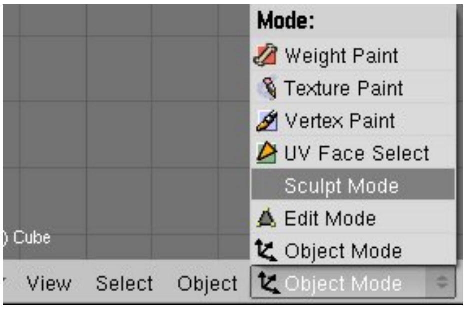
Previous: Manual/Subsurf Modifier	Contents	*Next:* Manual/SculptMode

User Manual: Contents | Guidelines | Blender Version **2.43**

Overview

Sculpt Mode is similar to Edit Mode in that it is used to alter the shape of a model, but Sculpt Mode uses a very different workflow: instead of dealing with individual elements (vertices, edges, and faces), an area of the model is altered using a brush. In other words, instead of selecting a group of vertices, Sculpt Mode automatically selects vertices based on where the brush is, and modifies them accordingly.

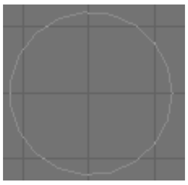
Sculpt Mode



Sculpt Mode Dropdown

Sculpt mode is selected from the mode menu.

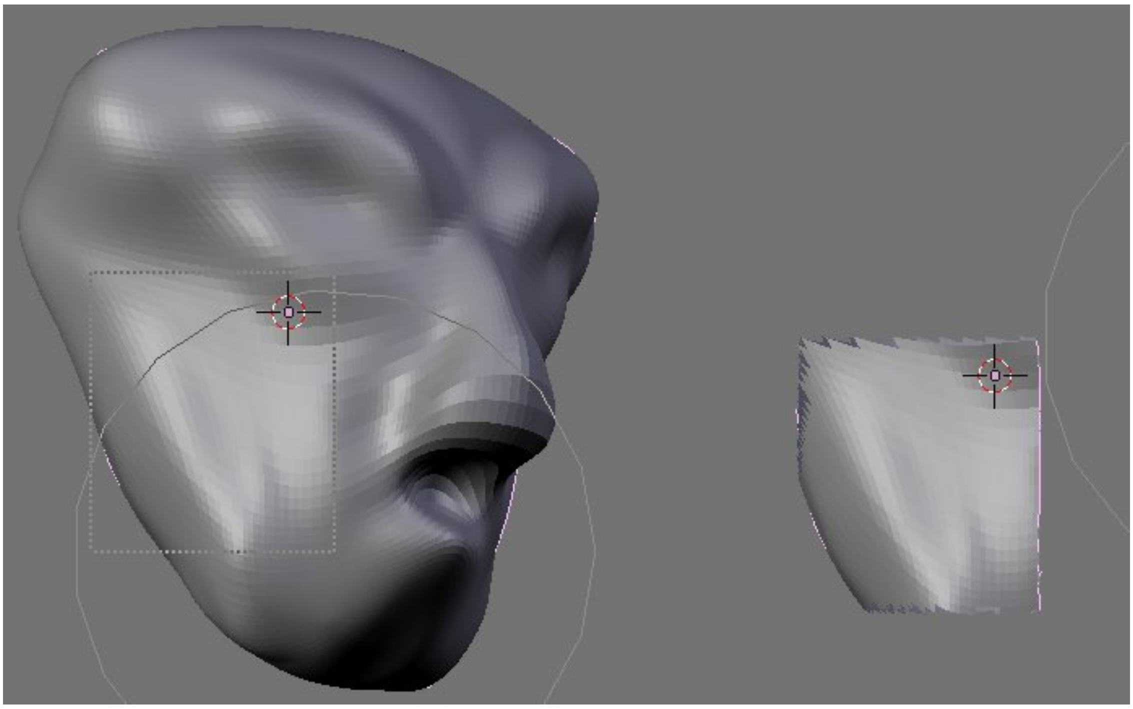
Once sculpt mode is activated a sculpt menu will appear in th 3d view header, and a tabs for sculpt and brush will appear in the multires panel. Also the cursor will change to a circle with a cross hair in the center.



The cursor in Sculpt Mode.

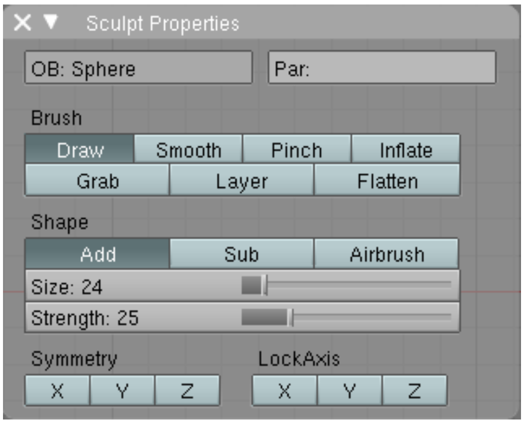
Hiding and Revealing Mesh

To hide mesh Shift Ctrl LMB drag will hide all but the selected rectangle. Or To Shift Ctrl RMB drag will hide only the selected rectangle. To reveal mesh Alt H or Shift Ctrl LMB click and release will reveal all of the mesh.



Hide before and after

Sculpt Panel

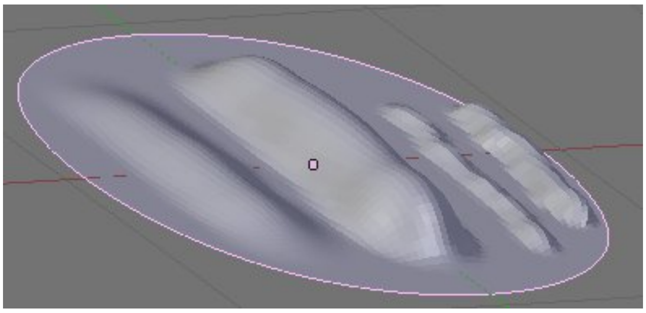


The Sculpt Properties panel.

This panel is opened with N or through the 3D view menu: Sculpt > Sculpt Properties. It is the exact equivalent of the Sculpt tab in the Edit buttons (F9).

Brushes

Sculpt Mode has seven brushes that each operate on the model in a unique way:



Drawing in various sizes and strengths.

Draw
: Draws a smooth curve on the model following the brush; vertices are displaced in the direction of the average normal of the vertices contained within the brush. D

Smooth
: As the name suggests, eliminates irregularities in the area of the mesh within the brush's influence. S

Pinch
: Pinch pulls vertices towards the center of the brush. If Sub is active instead of Add, vertices are pushed away from the center of the brush. P

Inflate
: Similar to Draw, except that vertices in Inflate mode are displaced in the direction of their own normals. I

Grab
: Grab is used to drag a group of points around. Unlike the other brushes, Grab does not modify different points as the brush is dragged across the model. Instead, Grab selects a group of vertices on mousedown, and pulls them to follow the mouse. The effect is similar to moving a group of vertices in editmode with proportional−editing enabled, except that Grab can make use of other Sculpt Mode options (like textures and symmetry.) G

Layer
: This brush is similar to Draw, except that the height of the displacement layer is capped. This creates the appearance of a solid layer being drawn. This brush does not draw on top of itself; brush stroke intersects itself. Releasing the mouse button and starting a new stroke will reset the depth and paint on top of the previous stroke. L

Flatten
: The Flatten will lower the height of the part of the mesh being worked on. Simply put, the direction of the flattening depends on the way the surface normals in the mesh are facing.

Modifiers

Brush Shape

Par
: The parent object's name.

Add and Sub
: Add causes the brush to pull an area of the model in the positive direction, Sub in the negative direction. (With the Pinch brush, Add pulls vertices inward and Sub pushes vertices outward.) Interactive toggling of brush direction is with holding down Shift. Or V can be used to toggle it until it is toggled again.

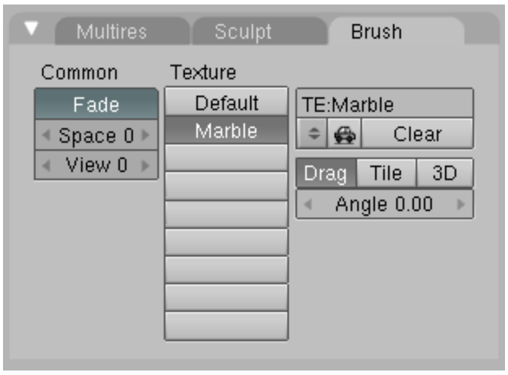
Airbrush
: When enabled, this option causes the brush to continue modifying the model after mouse down without moving the mouse. If disabled, the brush only modifies the model when the brush changes its location. A

Size
: This option controls the radius of the brush, measured in pixels. F in the 3D view allows you to change the brush size interactively by dragging the mouse and then left clicking (The texture of the brush should be visible inside the circle). Typing a number then enter while in F sizing allows you to enter the size numerically.

Strength
: Strength controls how much each application of the brush affects the model. For example, higher values cause the Draw brush to add depth to the model more quickly, and cause the Smooth brush to smooth the model more quickly. If the range of strengths doesn't seem to fit the model (for example, if even the lowest strength setting still makes too large of a change on the model) then you can scale the model (in Edit Mode, not Object Mode.) Larger sizes will make the brush's effect smaller, and vice versa. You can change the brush strength interactively by pressing Shift F in the 3D view and then moving the brush and then left clicking. You can enter the size numerically also while in Shift F sizing.

Symmetry
: Mirror the brush across the selected axes. Note that if you want to alter the directions the axes point in, you must rotate the model in Edit Mode, not Object Mode. Can be toggled via X, Y, and, Z respectively.

Brush Panel



The Brush panel.

Sculpt Mode can take full advantage of the range of options offered by Blender's texture system. Brush textures are accessed using a similar interface to that used by the Material buttons or the World buttons: there are nine texture slots in the Sculpt Mode Brush Panel, plus a Default slot that acts simply as a flat texture.

Any texture type can be loaded into one of the Sculpt Mode texture slots. Once a texture is associated with a slot, additional options will appear that affect how the texture controls the brush.

Drag, Tile and 3D
: These three options control how the texture is mapped onto the brush. If Drag is enabled, the texture follows the mouse, so it appears that the texture is being dragged across the model. The Tile option tiles the texture across the screen, so moving the brush appears to move seperately from the texture. The Tile option is most useful with tilable images, rather than procedural textures. Lastly, the 3D allows the brush to take full advantage of procedural textures. This mode uses vertex coordinates rather than the brush location to determine what area of the texture to use.

Fade

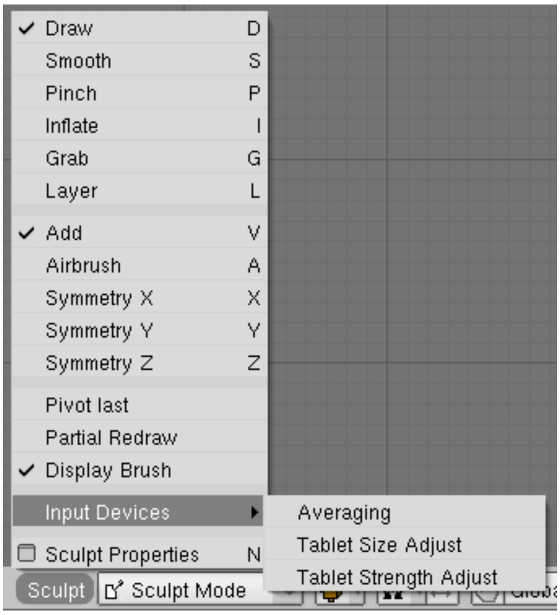
When Fade is enabled, this option smooths the edges of the brush texture, so that the brush will smoothly fade into the model at the edge of the brush's influence.

Space
: Setting this to a non−zero value adds extra space between each application of the brush. The value is measured in pixels; setting Space to 100 will require the mouse to move 100 pixels between each 'dot' applied to the mesh. Note that this is the total distance the brush has traveled, not the current linear distance from the last time the brush was applied.

View
: Pulls the brush direction towards view.

Angle
: This is the rotation angle of the texture brush. It can be changed interactively via Ctrl F in the 3D view. While in the interactive rotation you can enter a value numerically as well.

Sculpt Menu



The sculpt menu.

The Sculpt menu offers several new controls in addition to the tools already discussed.

Pivot last
: Sets the rotation center for rotating the scene to the last location the brush was used.

Partial Redraw
: This uses a special graphics optimization that only redraws where the mouse has been − it can speed up drawing on some graphics cards but slow it down on others. Primarily is only useful needed for dense mesh (greater than 100,000 polys).

Display Brush
: Controls whether the brush circle is drawn.

Input Devices
: Here you can select the behaviour of the used input devices.
    ◊ Averaging − This option uses an average direction of movement for the number of pixels specificed and then interpolates the stroke along a linear path of that number of pixels. This can be useful for dense meshes but the speed hit can be such that it may be faster to leave it at 1 (the default).

◊ Tablet Size Adjust − Sets to what extent tablet pressure affects brush size.
◊ Tablet Strength Adjust − Sets to what extent tablet pressure affects brush strength.

Keyboard Shortcuts

Action	**Shortcut**
Hide mesh outside selection	Shift Ctrl LMB
Hide mesh inside selection	Shift Ctrl RMB
Show entire mesh	Alt H
Toggle airbrush	A
Interactively set brush size	F
Interactively set brush strength	Shift F
Interactively rotate brush texture	Ctrl F
Toggle brush direction	V
Draw brush	D
Smooth brush	S
Pinch brush	P
Inflate brush	I
Flatten brush	T
Grab brush	G
Layer brush	L
X Symmetry	X
Y Symmetry	Y
Z Symmetry	Z
Toggle floating sculpt panel	N
Step up one multires level	Page Up
Step down one multires level	Page Down

Previous: Manual/Multires Contents *Next:* Manual/Retopo

User Manual: Contents | Guidelines | Blender Version **2.43**

Overview

Retopo (remake topology) is a tool for remaking the topology of a mesh. It's the opposite of the sculpting tools: instead of reshaping the model but leaving the topology the same, it reforms the topology, but maintains the same shape as the original model. You will not change the geometry of the original mesh in any way, you will be creating a new mesh that is projected upon an existing mesh. There are three ways to use retopo − retopo paint where you paint lines on the mesh and painted intersections become vertex locations; or by creating new mesh via standard mesh editing methods; or by projecting an existing mesh onto the object. In practice one usually uses all three together.

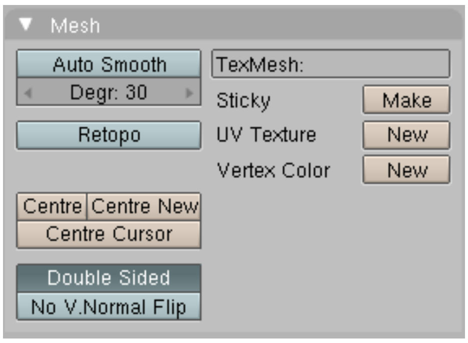
Usage

Retopo is controlled from the Mesh palette in the Editing buttons (F9).

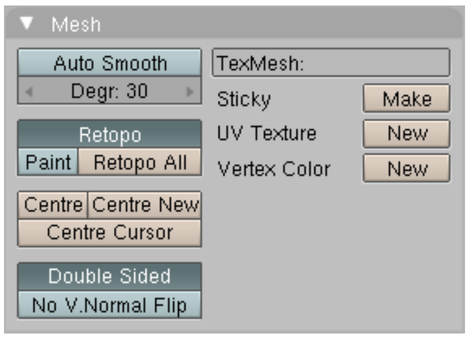
The Retopo button appears when a mesh is selected and EDIT mode is active.

When the Retopo toggle button is pressed, any change to a vertex will cause it to be snapped to the surface of *another* model. Note that this effect is view dependent: from the view you are working in, the vertices won't appear to have moved, because they are falling straight back along the axis until they hit the surface of the model.

Retopo Paint Overview



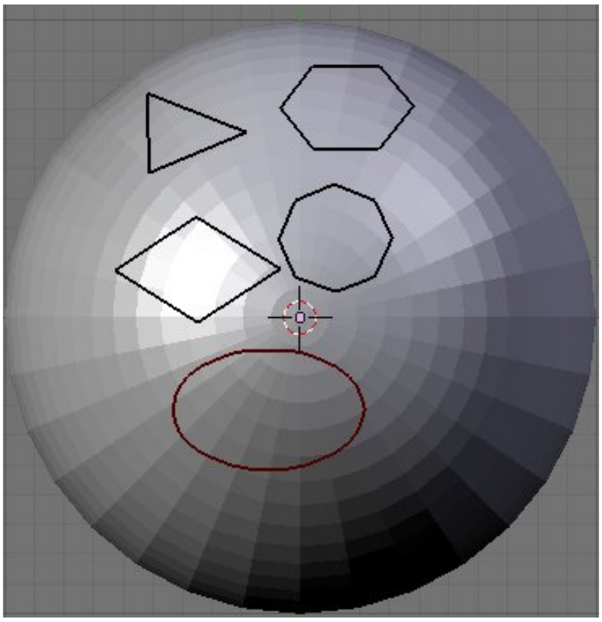
Mesh panel with Retopo disabled.



Mesh panel with Retopo enabled.

Once you click Retopo and then paint you will see the following options in the header.

The Retopo header.



Drawing ellipses.

Pen
: A freehand line that exactly follows the location of the mouse from your start click (LMB) until you release the mouse button.

Line
: A line segment from your start click (LMB) to where you release the mouse button.

Ellipse
: Draws a ellipse between a rectangular area with its center defined by your start click (LMB) and a corner defined by where you release the mouse.

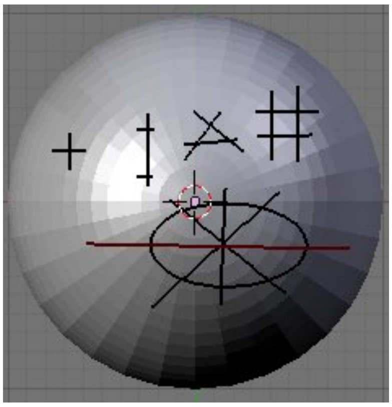
LineDiv and EllDiv
: LineDiv and EllDiv determine the number of segments in the line and ellipse line respectively. Thus a setting of 4 for EllDiv results in a diamond being draw instead of an ellipse. For LineDiv it determines the number of points used to project the line onto the object thus with two points only the endpoints are used, and the rest of the line does not follow the surface but instead passes through the object. Thus higher levels of LineDiv mean the projection will more accurately fit on the objects surface.

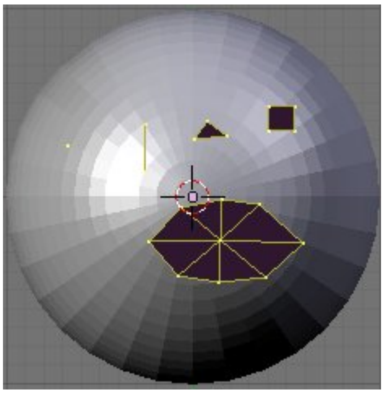
To extend a previous stroke go to an endpoint of the stroke and a circle will appear − click and drag to continue the stroke.

You can delete previous strokes by going to an endpoint (or for a circle along the edge) and when the circle appears click to turn the stroke red then Xkey or Delete to delete it.

Sometimes the option to extend can be an annoyance if you are trying to paint a new stroke near the endpoint of an existing stroke. In which case you can toggle off the hotspot for end point extension with the Hkey.

After painting your ellipses and lines press Enter to convert it to mesh. At each intersection of lines a vertice is formed. Two intersections on the same line forms an edge also. Three or four edges in a loop forms a face.

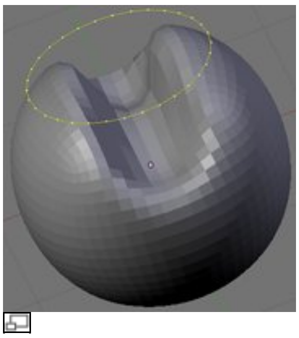

Drawing different shapes (mouse strokes).

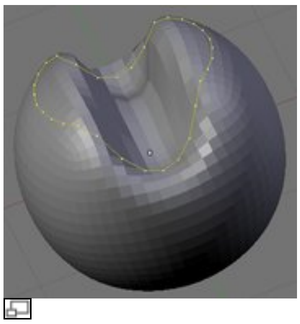

Drawing different shapes (results).

Limitations / Notes

- When you press Enter the mesh faces are derived based on your current view; thus you can only paint one side at a time.
- Lines cannot self intersect
- Can only autofill quads and tris, if an area has more than four surrounding vertices then a face will not be created
- New mesh from paint and previous mesh from paint must be stitched together manually.
- Each stroke is sequential.
- Paint strokes can only be viewed and edited in the 3D view you begin retopo paint in.

Retopo Mesh Overview


Circle before projection.


Circle projected onto carved sphere.

If you use Retopo without Paint you can use standard mesh editing tools to either create a new mesh. Or alternatively project an existing mesh onto the surface you wish to remake the topology of.

To project an existing mesh select the vertices you wish to project in the view you plan to project from then press Retopo All.

You may wish to hide the vertices you have already projected in order to avoid accidentally re−projecting them from a different view.

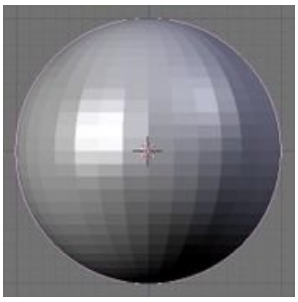
Limitations

- In order to avoid effecting mesh on the backside of the object you must hide the mesh (i.e. selection is not 'clipped' even if it is not visible).
- To re−topologize part of your current mesh you must either duplicate the entire object in object mode

or duplicate and separate part of the object in EditMode.

Tutorial

Step One



Step One

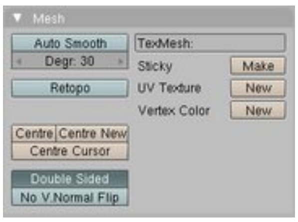
Since retopo modifies topology, not shape, we must first create the shape we will be using. This shape can be almost any 3D object: a mesh, a NURBS surface, metaballs, or even bones. One useful workflow is to quickly block out a shape in sculptmode, then use retopo on that mesh. For this example, I've chosen a simple UVsphere.

Step Two

Step Two

Add a new mesh. It doesn't matter what you choose; after you create it, press X to erase all vertices.

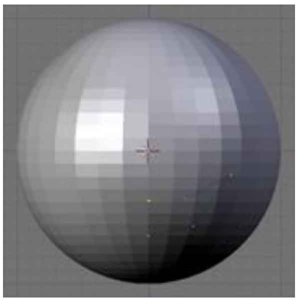
Step Three



Step Three

Turn on Retopo. The Retopo toggle is in the Mesh palette in the Editing buttons (F9). You should also check that you have Viewport Shading set to Solid and that Limit selection to visible is off (that's the cube icon next to the vertex/edge/face selection buttons.)

Step Four



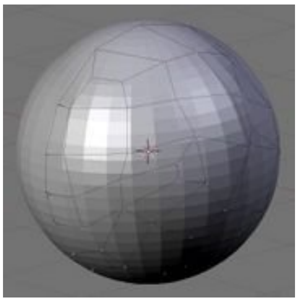
Step Four

Start adding points by clicking the left mouse LMB button while holding Ctrl down. This is a normal EditMode operation, except that if you now rotate your view, you can see that the vertices you just added are on the surface of the UVsphere. You can also use extrude, duplicate, grab, rotate, and scale; all of these operations will continue to snap vertices to the surface of the UVsphere.

Tip

To make using Retopo easier, make sure you're taking advantage of Blender's theme settings. You can use them to increase the size of vertices or to give better contrast to vertices and edges. You may also find it helpful to turn on the X−ray button in the Object buttons (F7 −> Draw panel). You may also find it useful to have multiple 3D viewports open so that you can see whether the vertex placement is as you desired without needing to rotate the model.

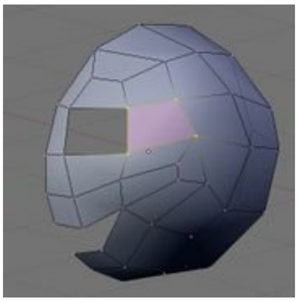
Step Five



Step Five

Continue adding points around the rest of the model. Make sure to connect vertices with edges so that you can see the topology you are creating.

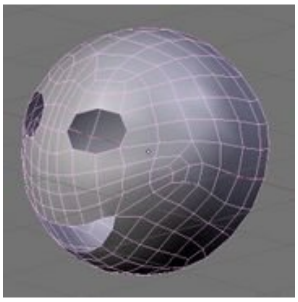
Step Six



Step Six

Once you have all the vertices and edges created, you can turn off Retopo and hide the UVsphere. Still in EditMode, start selecting groups of edges and use F to add faces. When you're done, you should have a complete surface.

Step Seven



Step Seven

Add a mirror modifier and, optionally, subsurf.

Previous: Manual/SculptMode	Contents	*Next:* Manual/Curves

User Manual: Contents | Guidelines | Blender Version **2.40**

Curves

Curves and Surfaces are objects just as meshes are objects except they are expressed in terms of mathematical functions, rather than as a series of points.

Blender implements both Bézier and Non Uniform Rational B−Splines (NURBS) curves and surfaces. Both are defined in terms of a set of "control vertices" which define a "control polygon", though each follow a different set of mathematical laws. The way the curve and the surface are interpolated might seem similar, at first glance, to Catmull−Clark subdivision surfaces. The curve is *interpolated* while the surface is *attracted*.

When compared to meshes, curves and surfaces have both advantages and disadvantages. Because curves are defined by less data, they produce nice results using less memory at modelling time, whereas the demands increase at rendering time.

Some modelling techniques, such as extruding a profile along a path, are only possible with curves. But the very fine control available on a per−vertex basis on a mesh, is not possible with curves.

There are times when curves and surfaces are more advantageous than meshes, and times when meshes are more useful. If you have read the chapter on Basic Mesh Modelling and Advanced Mesh Modelling, and you read this chapter, you will be able to choose whether to use meshes or curves.

Logo Thumbnail

Working with curves in Blender is fairly simple and surprisingly there are very few HotKeys when creating curves. It is what you do with those curves that really makes the difference. A curve by itself is just that, a curve. But a curve applied to another curve can create very complex objects.

When you have finished reading and learning about Bézier and NURBS curves there are several more advanced examples on the application of curves in the tutorials section for modelling complex objects.

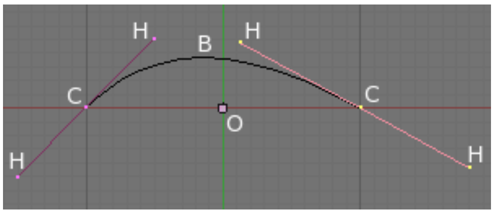
There is a Working example that shows how to create an interesting bird−like logo, (*Logo*). The tutorial covers most aspects of working with Bézier curves including: adding curves, setting up a background image as a template guide and beveling the final curve.

In addition, the Tutorial section has examples on both Skinning and Curve deform techniques.

BÃ©ziers

BÃ©zier curves are the most commonly used curve for designing letters or logos. They are also widely used in animation, both as paths for objects to move along and as IPO curves to change the properties of objects as a function of time.

There are three panels designed to assist in working with and modifying curves: Curve and Surface, Curve Tools and Curve Tools1. Each panel has buttons that change the characteristics of curves.

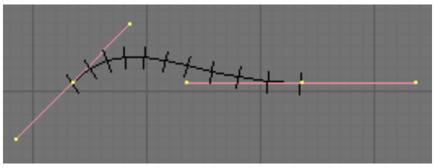


Curve example.

(*Curve example*) is the most basic curve you can create. It consists of two control points or vertices, labeled "**C**", the curve "**B**", handles "**H**"s and an object center "**O**".

Selecting the control point also selects the handles, and allows you to move the complete vertex. Selecting one or more handles allows you to change the shape of the curve by dragging the handles.

To create a curve use the Toolbox's *Add/Curve/BÃ©zier Curve* menu entry to add a new curve, (*Curve example*). By default the new curve exists only in 2D. For example, if you created the curve in the Top view, the shape of the curve can only be change in the XY Plane. You can apply transforms to the curve but you can't change its shape in 3D.



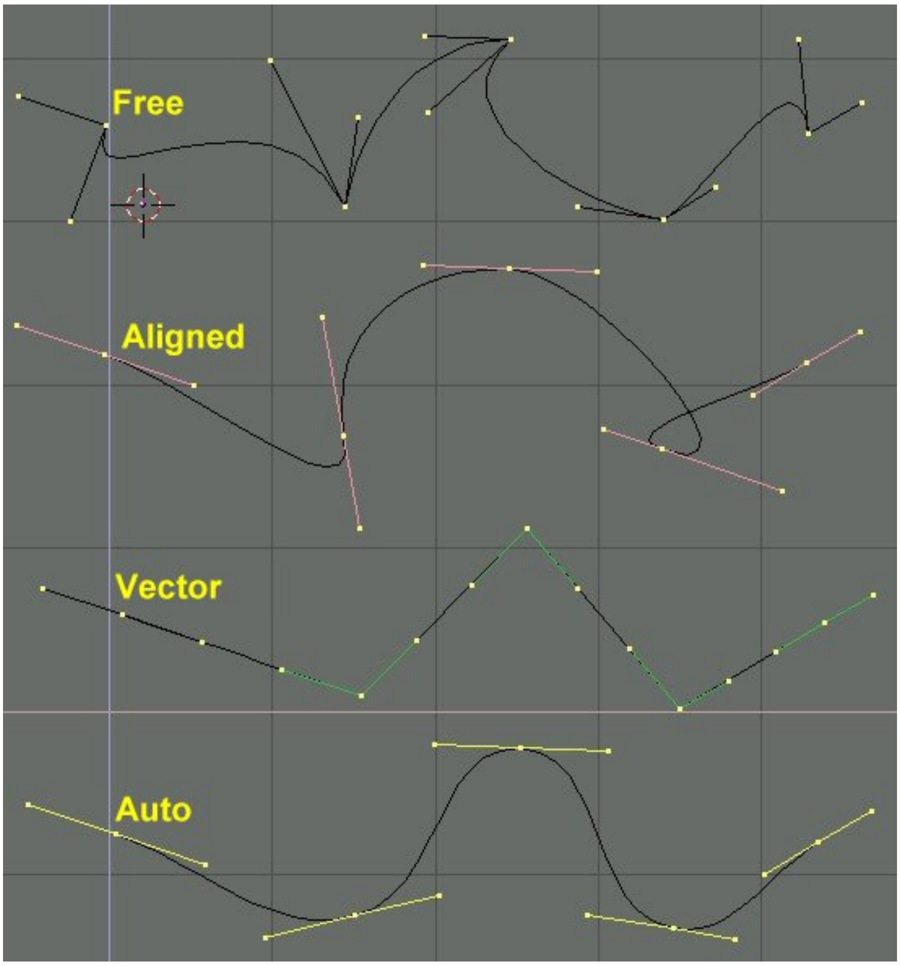
3D Curve − a Path

To work with the curve in 3D you need to *turn on* the 3D property of the curve using the **3D** button on the Curve and Surface panel. You can visually see that a curve is in 3D by noticing the curve has *railroad* tracks or marks. (*3D Curve*) is a 3D curve and (*Curve example*) is a 2D curve.

A handle is always tangent to the curve. The 'steepness' of the curve is controlled by the handle's length, any "**H**" to a "**C**". The longer a handle is the steeper the curve (i.e. the more curve wants to *hug* the handle).

There are four types of handles (*Types of Handles for BÃ©zier curves*):

- Free Handle (black). The handles are independent of each other. To convert to Free handles use H. H also toggles between Free and Aligned.
- Aligned Handle (purple). These handles always lie in a straight line. Hotkey: H (toggles between Free and Aligned). The curve enters and exits the control point along handles.
- Vector Handle (green). Both parts of a handle always point to the previous handle or the next handle. Hotkey: V;
- Auto Handle (yellow). This handle has a completely automatic length and direction, set by Blender to ensure the smoothest result. Hotkey: Shift H.



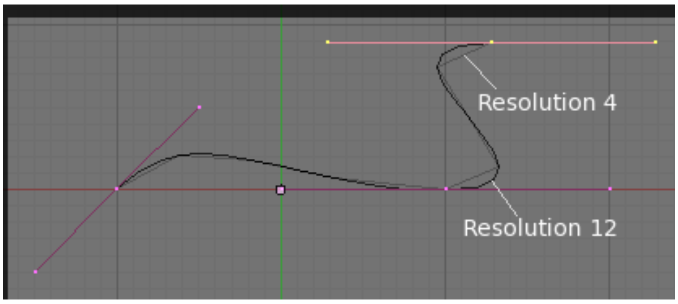
Types of Handles for Bézier curves

Handles can be *grabbed*, *rotated* and *scaled* exactly as ordinary vertices in a mesh would. As soon as the handles are moved, the handle type is modified automatically:

- Auto Handles becomes Aligned;
- Vector Handles becomes Free.

Curve resolution

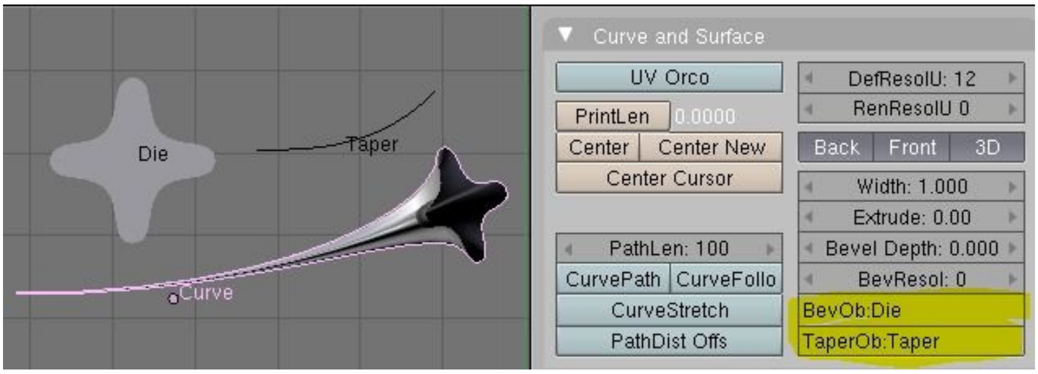
Although the BÃ©zier curve is a continuous mathematical object it must nevertheless be represented in discrete form from a rendering point of view. This is done by setting a *resolution* property, which defines the number of points which are computed between every pair of control points.



Resolution example

A separate *resolution* can be set for each BÃ©zier curve by adjusting the DefResolIU Field. The default is **6**. (*Resolution example*) is an example of the same curve, superimposed, with the aid of *Gimp*, showing two different resolution settings. The lighter shaded curve has a low resolution of "**4**"; the curve begins to look linear. The darker curve has a resolution of "**12**" and is very smooth. Note: high resolutions may look nicer but they can slow down interactive rendering if there is a large number of curves.

Bevel and Taper Objects



3D Curve modified by Bevel and Taper curves

A Bevel object, applied to a Curve object, forms a skin for the curve. Where the curve is the path or length of a pipe, the Bevel Object defines the outside shape, like the outside of a cord, or hose pipe. Normally the Bevel is a simple round circle, and thus makes the curve into a pipe or soda can. The Bevel shape must be two−dimensional, and it can be rectangular for simulating wrought iron or flat steel, oval (with a crease) for a power cord, star−shaped for a shooting star illustration; anything that can be physically formed by extrusion (extruded).

A Taper object is an open curve with control points above its object center. When applied to a Beveled Curve, it changes the diameter of the Bevel along the length of the curve, like a python just having eaten a rat, or like a hose bulging up under pressure, or a vine growing.

For adjusting proper size of Bevel effect for individual curve's segment use Set Radius option accessable through W−4. Default value is 1.0.

Caution: no Bevel Effect if bevel Radius parameter set to 0.0.

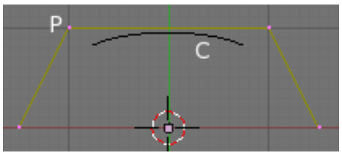
NURBS

NURBS curves are defined as rational polynomials and are more general, strictly speaking, than conventional B−Splines and Bézier curves inasmuch as they are able to exactly follow any contour. For example a Bézier circle is a polynomial *approximation* of a circle, and this approximation is noticeable, whereas a NURBS circle is *exactly* a circle.

NURBS curves require a little bit more understanding of the underlying components that make up a NURBS curve in order to get full use of them. They have a large set of variables, which allow you to create mathematically pure forms. However, working with them requires a little more discussion on the various parts of a NURBS curve.

Uniform−Endpoints

We start with *Knots*. NURBS curves have a *knot* vector, a row of numbers that specifies the parametric definition of the curve (i.e. they describe the range of influence for each of the control−points). Remember the control−points from Bézier curves, NURBS have them too and each control−point affects some part of the curve along that range. The control−points appear as purple vertices.

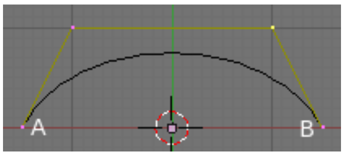


Default Uniform curve

(*Default Uniform curve*) is the default NURBS curve created using the "NURBS Curve" menu item from the Toolbox's Add menu and is an example of a Uniform curve. The curve itself is drawn in black, labeled "**C**" and the control−points are drawn in purple; one out of the 4 is labeled "**P**".

You can't manipulate the *Knot* vector directly but you can configure it using two pre−sets: Uniform and Endpoint.

The Uniform button produces a uniform division for closed curves, but when used with open curves you will get "free" ends, which are difficult to locate precisely.



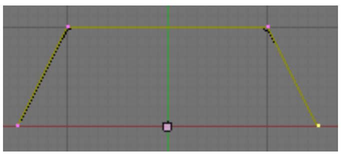
Endpoint curve

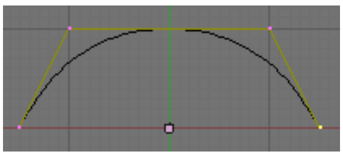
The Endpoint button sets the *Knot* vector in such a way that the first and last vertices are always part of the curve, which makes them much easier to place. (*Endpoint curve*) is an example of applying the Endpoint button to the (*Default Uniform curve*). You can see that the curve has now been pulled to the *end* control−points labeled "**A**" and "**B**".

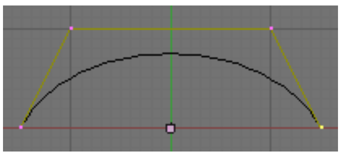
Order

The Order field is the 'depth' or *degree* of the curve (i.e. you are specifying how much the control−points are taken into account for calculating the curve shape).

Order **1** is a point and is not an available depth setting, Order **2** is linear (*Order 2 curve*), Order **3** is quadratic (*Order 3 curve*), (*Order 4 curve*) and so on. The valid range is "**2**" to "**6**". Notice that as the Order rises the curve moves away from the control−points.


Order 2 curve.


Order 3 curve.


Order 4 curve.

If your curve has **6** or more control−points the Order can not be set higher than **6**. **6** is the highest Order

allowable. If you have less than **6** control−points then the highest Order is limited by the number of control−points. For example, if your curve has **5** control−points then the highest Order allowable is **5**.

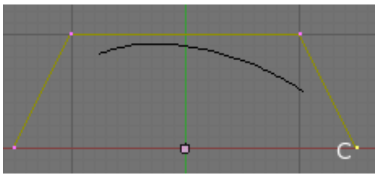
Always use an order of **5**, if possible, for curve paths because it behaves fluidly under all circumstances, without producing irritating discontinuities in the movement. For example, if you have a cube assigned to travel along a NURBS path with an Order of say **2** then the cube will appear to move roughly (or jerky) along the path.

Math Note

Mathematically speaking the Order is the order of both the Numerator and the Denominator of the rational polynomial defining the NURBS curve.

Weight

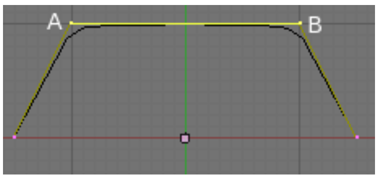
NURBS curves have a Weight assigned to each control−point that controls how much each "pulls" on the curve. Think of it as if each control−point has an arm that reaches out and grabs hold of the curve and tries to pull on it. The larger the Weight the more the control−point pulls on the curve, see (*Weight of 5*) and (*Weight of 20*). The valid range of Weight settings are **0.1** to **100.0**.



Weight of 5.

Weight of 20.

The larger Weight of 20 pulls the curve towards the control−point labeled "**C**". Each control−point can have a different Weight setting. As the Weight for a control−point increases the curve will hug the control−points closer. If the Weights are large enough the curve will almost follow the control−points, see (*Weight of 100*).



Weight of 100.

The control−points can effectively compete with each other. For example, the control−point with the largest Weight will pull the curve towards it and away from the others. If all the control−points have the same Weight then the Weight is effectively canceled, as if none had Weights.

In (*Weight of 100*) the top two control−points have their Weight set at **100.0**, labeled "**A**" and "**B**". The opposite control−points have their Weight at **1.0**. You can see that the curve is "pulled" toward control−points "**A**" and "**B**". And at such a high Weight the curve almost follows the control−points.

On the Weight panel there are four preset Weights that provide typical Weight settings for certain kinds of control−point arrangments. Some generate Weight settings that are used for control−points that form circles.

To see the Weight value of a control−point open the Transform Properties panel using N and look at the Vertex W field. The Weight field doesn't show the Weight.

Resolution

As with Béziers curves NURBS curves' Resolution can be controlled from the Curve Tools panel.

Opening−Closing−Deleting−Joining−Bevel−Taper

As with Béziers curves, Opening, Closing, Deleting and Joining NURBS curves is performed using the same Hotkeys and same Curve Tools, with the same rules applying, see Béziers curves section.

Previous: Manual/Retopo	Contents	*Next:* Manual/Editing Curves

Add new segment

Mode: Edit mode

Hotkey: Ctrl LMB

Menu: Curve â†’ Extrude

Description

Once a curve is created you can add new segments by extruding it. Each new segment is added to the end of the curve. A new segment will only be added if a single vertex, or handle, at one end of the curve is selected. If two or more vertices are selected nothing is added.

Opening and Closing a Curve

Mode: Edit mode

Hotkey: C

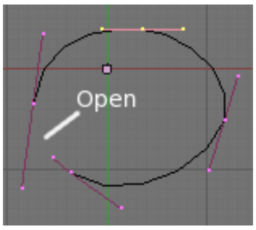
Menu: Curve â†’ Toggle Cyclic

Description

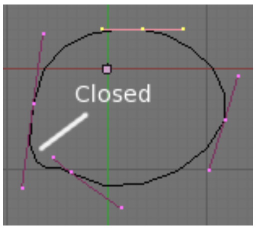
This toggles between an open curve and closed curve. The shape of the closing segment is based on the handles. The only time a handle is adjusted after closing is if the handle is an Auto handle. (*Open curve*) and (*Closed curve*) is the same curve open and closed.

This action only works on the original starting control−point or the last control−point added. Deleting a segment(s) dosen't change how the action applies; it still operates only on the starting and last control−points. This means that C may actually join two curves instead of closing a single curve.

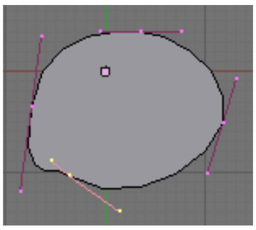
Examples



Open curve.



Closed curve.


Closed curve (Solid).

If the curve is closed the curve is automatically considered a surface with an area. This means it is rendered as a solid (*Closed curve (Solid)*) and it is renderable using F12.

Deleting a Segment(s)

Mode: Edit mode

Hotkey: X

Menu: Curve â†’ Delete

Description

A segment is defined implicitly by selecting two adjacent control−points. You can't explicitly select a segment; you must select two adjacent control−points. Once the control points are selected you can use the Erase/Delete menu and selecting the *Segment* menu item.

Hints

You can delete multiple segments by selecting one or more control−points or handles. Use the Erase/Delete menu and select *Selected*.

Joining two curves

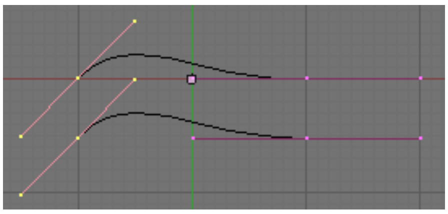
Mode: Edit mode

Hotkey: F

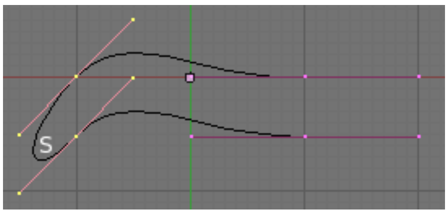
Menu: Curve â†’ Make Segment

Description

Joining curves is really the act of making a segment between the two curves. To join two seperate use one control point from each curve. The two curves are joined by a segment to become a single curve. (*One curve joined*) is the result of joining (*Two curves*). The segment, labeled "**S**", is the new segment joining the two curves. We use F for the hotkey because it is similar to making a new **F**ace in a mesh.



Two curves.



One curve joined.

You can not close a curve by joining the curves; you must Close the curve. You will get the error "**Can't make segment**" when you attempt to join using the starting and last control−point. For example, in (*One curve joined*) you must use Close to close the curve.

Previous: Manual/Curves Contents *Next:* Manual/Curves Deform

Curve Deform Section

This document was originally from a tutorial section. A copy of it was made and put in this section because the main index of the user manual for Blender did not seem cover Curve Deforming directly but only as a tutorial. This document is out of date and needs some corrections but should work for Curve Deforming.

Curve Deform

Curve Deform provides a simple but efficient method of defining a deformation on a mesh. By parenting a mesh object to a curve, you can deform the mesh up or down the curve by moving the mesh along, or orthogonal to, the dominant axis.

The Curve Deform works on a dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction, the mesh will traverse along the curve. Moving the mesh in an orthogonal direction will move the mesh object closer or further away from the curve. The default settings in Blender map the Y axis to the dominant axis. When you move the object beyond the curve endings the object will continue to deform based on the direction vector of the curve endings.

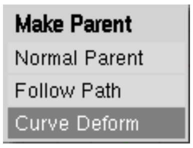
A Tip

Try to position your object over the curve immediately after you have added it, before adding the curve deform. This gives the best control over how the deformation works.

Interface

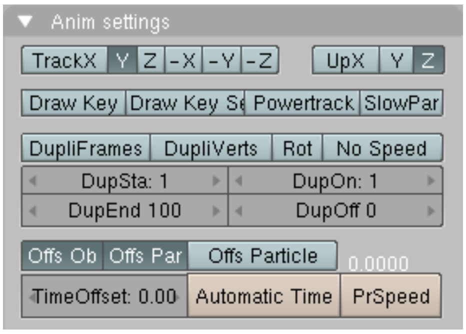
When parenting a mesh to a curve (CTRL−P), you will be presented with a menu, *Make Parent menu*.

By selecting Curve Deform you enable the Curve Deform function on the mesh object.



Make Parent menu.

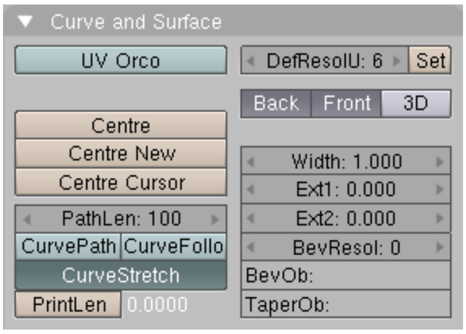
The dominant axis setting is set on the mesh object. By default the dominant axis in Blender is Y. This can be changed by selecting one of the Track X, Y or Z buttons in the Anim Panel, *Anim settings panel.*, in Object Context (F7).



Anim settings panel.

Cyclic curves work as expected where the object deformations traverse along the path in cycles.

CurveStretch provides an option to let the mesh object stretch, or squeeze, over the entire curve. This option is in Edit Context (F9) for the curve. See *Curve and Surface panel*.

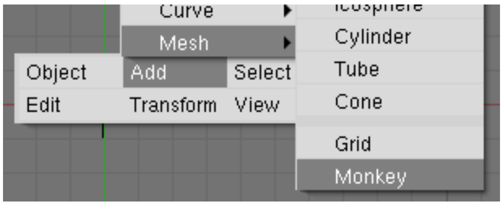


Curve and Surface panel.

Example

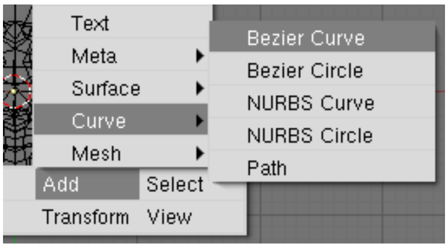
Let's make a simple example:

- Remove default cube object from scene and add a Monkey! (SHIFT−A −> Add −> Mesh −> Monkey, *Add a Monkey!*).



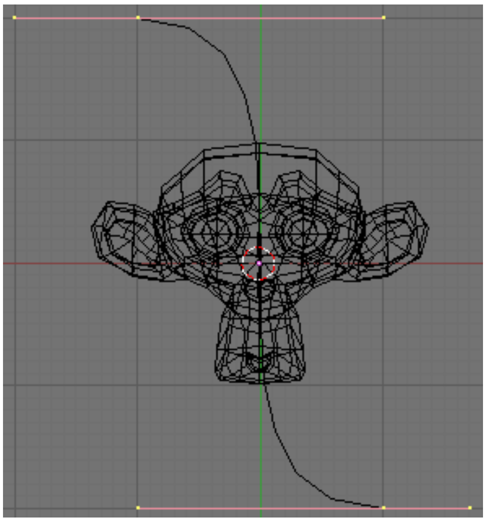
Add a Monkey!

- Now press TAB to exit Edit Mode.
- Now add a curve. (SHIFT−A −> Add −> Curve −> Bezier Curve, *Add a Curve*).



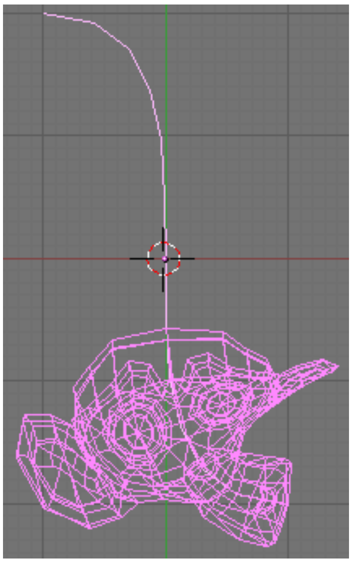
Add a Curve.

- While in Edit Mode, move the control points of the curve as shown in *Edit Curve.*, then exit EditMode, (TAB).



Edit Curve.

- Select the Monkey, (RMB), and then shift select the curve, (SHIFT−RMB).
- Press CTRL−P to open up the Make Parent menu.
- Select Curve Deform. (*Make Parent menu*).
- The Monkey should be positioned on the curve as (*Monkey on a Curve*).



Monkey on a Curve.

- Now if you select the Monkey, (RMB), and move it, (G), in the Y−direction, (the dominant axis by default), the monkey will deform nicely along the curve.

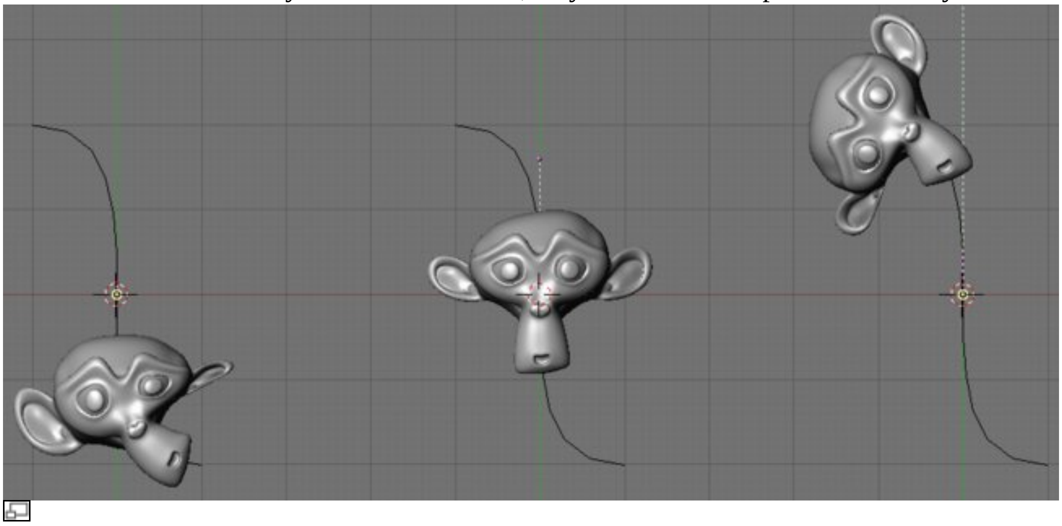
A Tip

If you press MMB while moving the Monkey you will constrain the movement to one axis only.

- In *Monkey deformations.*, you can see the Monkey at different positions along the curve. To get a cleaner view over the deformation I have activated SubSurf with Subdiv 2 and Set Smooth on the Monkey mesh. (F9 to get Edit options).

A Tip

Moving the Monkey in directions other than the dominant axis will create some odd deformations. Sometimes this is what you want to achieve, so you'll need to experiment and try it out!



Monkey deformations.

Previous: Manual/PartII/Advanced Mesh Modelling	Contents	*Next:* Manual/PartII/Surfaces

Use this navigator if you wish to follow the Blender user manual index flow:

Previous: Manual/Editing Curves	Contents	*Next:* Manual/Surfaces

User Manual: Contents | Guidelines | Blender Version **2.40**

Surfaces

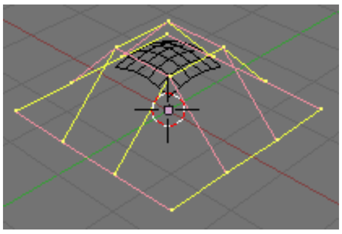
Surfaces are actually an extension of NURBS curves but are still a unique object unto themselves. Whereas a curve produces only one−dimensional interpolation, Surfaces have a second extra dimension of interpolation. The first dimension is U, as for curves, and the second dimension is V.

You may ask yourself "but the surface appears to be 3D, why is it only 2D?". In order to be 3D the object needs to have "Volume" and a surface doesn't have a volume, it is infinitely thin. If it has a volume the surface would have a thickness. Even though the surface appears to extend in 3D, it has no volume, and hence it only has two interpolation coordinates, U and V. **U** is the Yellow grid lines and **V** is the pink grid lines in (*Surface*).

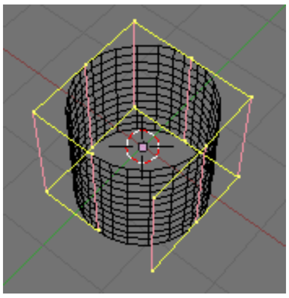
Many of the concepts from NURBS curves carry directly over to NURBS Surfaces, such as control−points, Order, Weight, Resolution etc..

For NURBS Surfaces the control−points form a grid and is sometimes called a "Cage". The grid performs exactly like the control−points of a NURBS curve; they control the boundary of the surface.

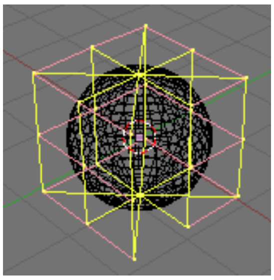
To help get started in creating surfaces there are four preset NURBS Surfaces: (*Surface*), (*Tube*), (*Sphere*) and (*Donut*).



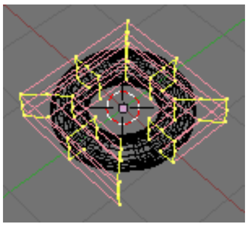
Surface.



Tube.



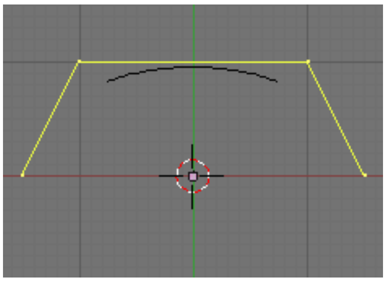
Sphere.

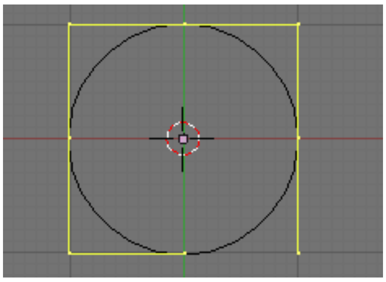


Donut.

Each preset is accessable from the Surface sub−menu of the Toolbox menu where each is designed as a starting point for creating more complex surfaces, of which the most common starting surface is (*Surface*).

There are also two preset NURBS Surface Curves: (*Curve*) and (*Circle*).


Curve.


Circle.

Although they visually look like NURBS curves they are *not*. Blender internally treats NURBS Surface Curves and NURBS Curves completely different. There are several attributes that seperate them but the most important is that a NURBS Curve has a single interpolation axis and a NURBS Surface Curve has two interpolation axes.

Visually you can tell which is which by entering Edit mode and looking at the 3D window's header; either the header shows "Surface" or "Curve" as one of the menu choices. Also, you can Extrude a NURBS Surface Curve to create a surface but you can't with a NURBS Curve.

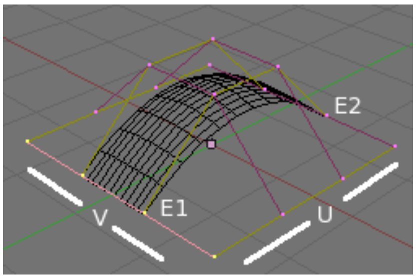
Use Surfaces to create and revise fluid curved surfaces. Surfaces can be cyclical in both directions, allowing you to easily create a Donut shape, and they can be drawn as 'solids' in Edit mode. This makes working with surfaces quite easy.

Note

Currently Blender has a basic tool set for Surfaces, with limited ability to create holes and melt surfaces. Future versions will contain increased functionality.

Uniform−Endpoints

Just like with NURBS curves, NURBS Surfaces have a knot vector and the configuration of the knot values are controlled by the Uniform and Endpoint buttons. Each interpolation axis can be independently set to either Uniform or Endpoint.



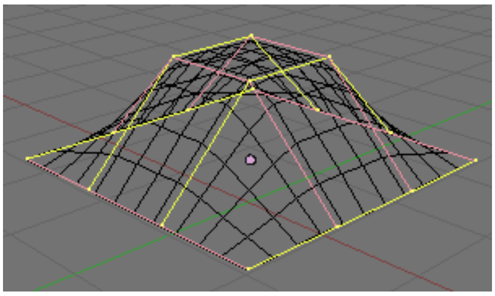
Endpoint U

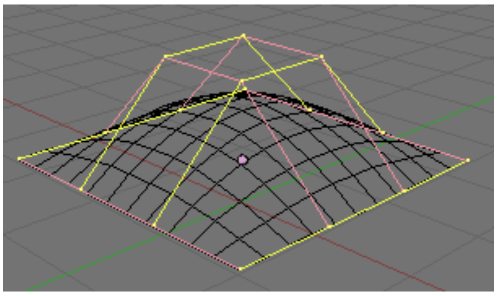
In (*Endpoint U*), the U interpolation axis is labeled as "**U**" and the V interpolation axis is labeled as "**V**". The **U**'s interpolation axis has been set to Endpoint and as such the surface now extends to the outer edges from "**E1**" to "**E2**" along the **U** interpolation axis.

To cause the surface to extend to all edges you would set the **V'**s axis to Endpoint as well.

Order

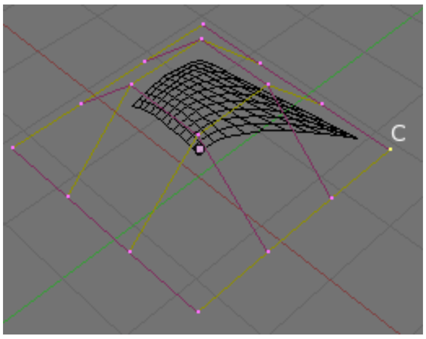
As with NURBS Curves, Order specifies how much the control−points are taken into account for calculating the curve of the surface shape. For high Orders, (*Order 4 surface*), the surface pulls away from the control−points creating a smoother surface; assuming that the Resolution is high enough. For low Orders , (*Order 2 surface*), the surface follows the control−points creating a surface that tends to follow the grid cage.


Order 2 surface.


Order 4 surface.

For illustration purposes, in both (*Order 4 surface*) and (*Order 2 surface*), the knot vectors were set to Endpoint causing the surface to extend to all edges.

Weight



Surface Weight 100

Again, as with NURBS Curves, Weight specifies how much each control−point "pulls" on the curve.

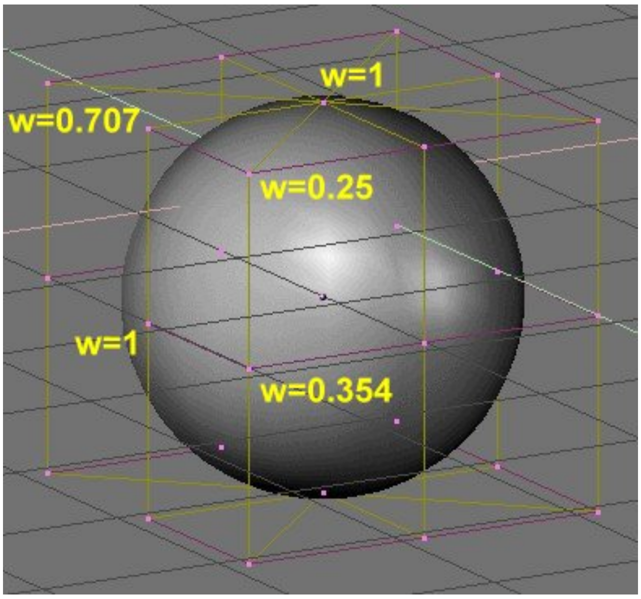
In (*Surface Weight 100*), a single control−point, labeled "**C**", has had its Weight set to 100.0 while all others are at their default of 1.0. As you can see that control−point *pulls* the surface towards it.

If all the control−points have the same Weight then each effectively cancels each other out. It is the difference in the Weights that cause the surface to move towards or away from a control−point.

The Weight of any particular control−point is visible in the Transform properties panel which is accessed using the N.

See NURBS Curves Weight for further details.

Preset Weights



A sphere surface

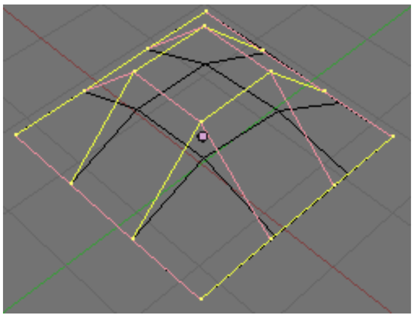
NURBS can create pure shapes such as circles, cylinders, and spheres (but note that a Bézier circle is not a pure circle.) To create pure circles, globes, or cylinders, you must set the weights of the control−points. This is not intuitive, and you should read more on NURBS before trying this.

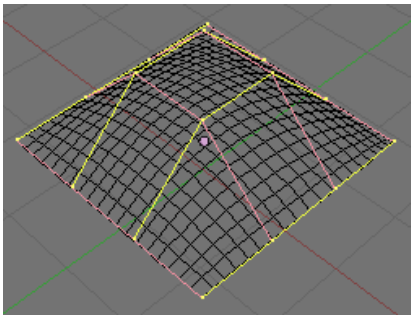
Basically, to produce a circular arc from a curve with three control−points, the end points must have a unitary weight, while the weight of the central control point must be equal to one−half the cosine of half the angle between the segments joining the points. (*A sphere surface*) shows this for a globe. Three standard numbers are included as presets in the Curve Tools panel.

Resolution

Just like NURBS Curves, Resolution controls the detail of the surface. The higher the Resolution the more detailed and smoother the surface is. The lower the Resolution the rougher the surface.

(*Resolution 4x4 surface*) is an example of a surface resolution of 4 for both **U** and **V**. (*Resolution 20x20 surface*) is an example of a surface resolution of 20 for both **U** and **V**.


Resolution 4x4 surface.


Resolution 20x20 surface.

For illustration purposes the knot vectors where set to Endpoint causing the surface to extend to all edges.

Adding or Extruding

Mode: Edit mode

Hotkey: E

Menu: Surface â†’ Extrude

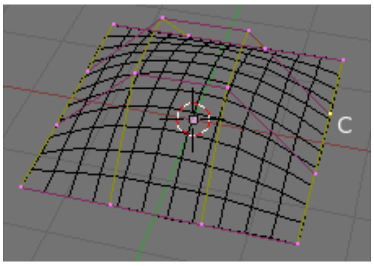
Description

Once the tool is activated the extrusion happens immediately and you are placed into Grab mode, ready to drag the new extruded surface to its destination.

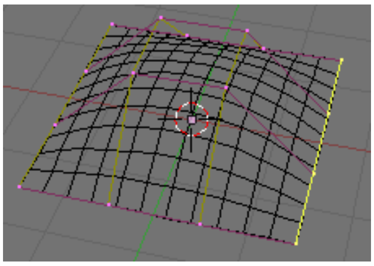
Examples

Images (*Selecting control−point*) to (*Complete*) show a typical extrusion along the side of a surface.

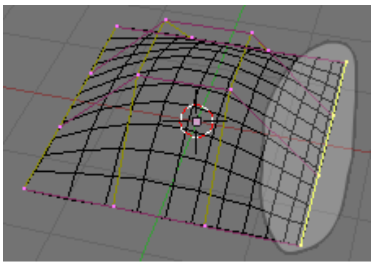
In (*Selecting control−point*) and (*Shift−R*), a row of control−points were highlighted by selecting a single control−point, labeled "**C**", and then using the handy row select tool (Shift R) to select the rest of the control−points.

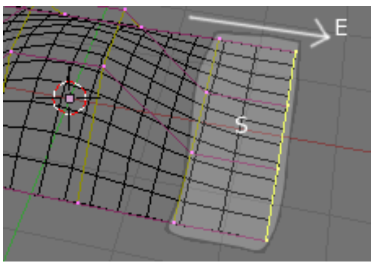


Selecting control−point.


Shift−R.

The edge is then extruded using the E as shown in (*Extruding*). Notice how the mesh has bunched up next to the highlighted edge; the area in question is highlighted in a light−grey cicular area. That is because the *new* extruded surface section is bunched up there as well.


Extruding.



Complete.

By moving the new section away from the area the surface begins to "unbunch", as shown in (*Complete*). The direction of movement is marked with a white arrow, labeled "**E**", and the new section is labeled "**S**".

You can continue this process of extruding −−or adding−− new surface sections until you have reached the final shape for your model.

Cycling (Opening and Closing)

Mode: Edit mode

Hotkey: C

Menu: Surface â†’ Toggle Cyclic

Description

Cycling a surface is similar to [Opening and Closing a NURBS curve] except that a surface has an inside and outside surface.

To cycle a surface use C and choose either "cyclic U" or cyclic V" from the [Toggle] menu. The surface's outer edges will join together to form a "closed" surface.

Attempting to cycle a non−outer edge will result in nothing happening.

Deleteing/Erasing surfaces

Mode: Edit mode

Hotkey: X

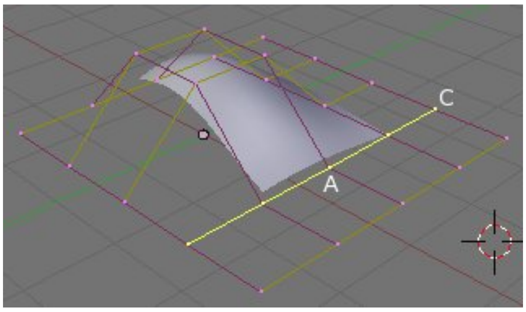
Menu: Curve â†’ Delete

Description

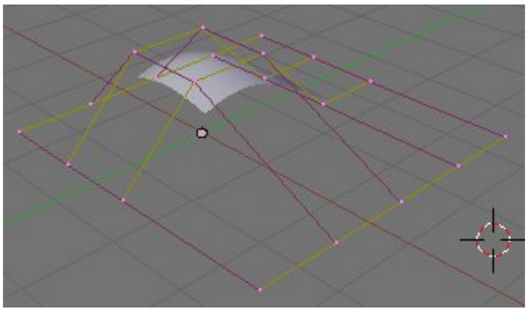
Deleting requires that all control−points along an interpolation axis are highlighted.

Hints

A handy Hotkey (Shift R) is provided that makes it easier to select all the control−points along an axis. Just highlight a control−point(s) and use Shift R to toggle between the two interpolation axes that intersect the last control−point selected.



Before.



After.

In (*Before*) a row of control−points have been selected by initially selecting the control−point labeled "**A**" and using Shift R to select the remaining control−points. Then, using the [Erase menu] (X), the *selected* row of

control−points is erased resulting in (*After*).

Joining or Merging two surfaces

Mode: Edit mode

Hotkey: F

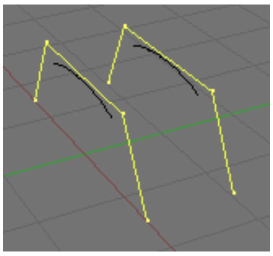
Menu: Surface â†’ Make Segment

Description

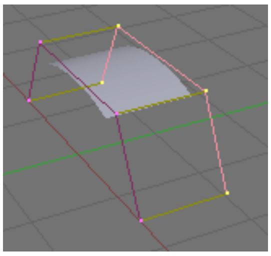
Just like NURBS Curves, Joining requires that a single edge, a row of control−points, from two seperate surfaces are selected. This means that the surfaces must be part of the same object. For example, you can't join two surfaces while in Object mode. Joining can only take place while in Edit mode which requires that both surfaces be part of the same object.

Examples

(*Joining ready*) is an example of two NURBS Surface curves, ***not*** NURBS curves, in Edit mode ready to be joined. (*Joining complete*) is the result of joining the two curves.



Joining ready.



Joining complete.

Hints

If not enough surfaces are selected then you will get an error message stating "Too few selections to merge".

Most of the time the Join tool will try its best to join the two surfaces based on the selected edges from those surfaces. But there are times when joining doesn't happen. Generally this occurs when the selected control−points are not completely describing the edge/row that you want to join. Select more control−points until the edge is completely highlighted.

Note that the edges do not have to be outside edges. You can join inside edges, although this is not typically done.

Previous: Manual/Surfaces	Contents	*Next:* Manual/Surfaces Skinning

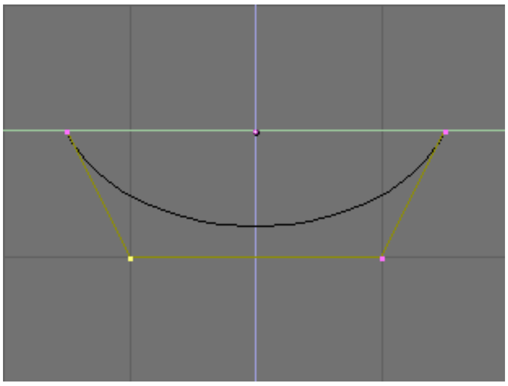
Skinning Section

This document was originally from a tutorial section. A copy of it was made and put in this section because the main index of the user manual for Blender did not seem cover skinning for surfaces/mesh/curves. This document is out of date and needs some corrections but should work for surface skinning.

Skinning

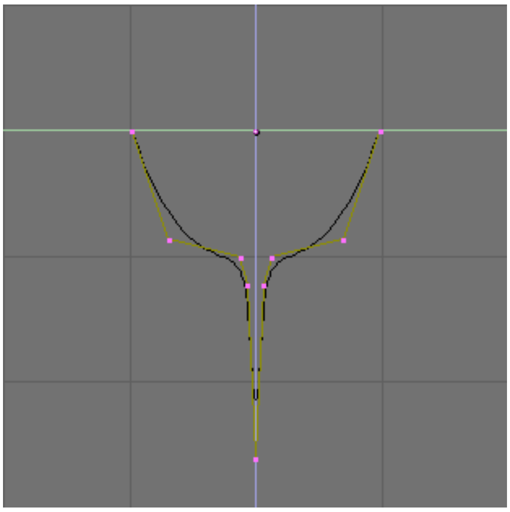
Skinning is the fine art of defining a surface using two or more profiles. In Blender you do so by preparing as many curves of the the desired shape and then converting them to a single NURBS surface.

As an example we will create a sailboat. The first thing to do, in side view (NUM3), is to add a Surface Curve. Be sure to add a *Surface* curve and not a curve of Bézier or NURBS flavour, or the trick won't work (*A Surface curve for skinning.*).



A Surface curve for skinning.

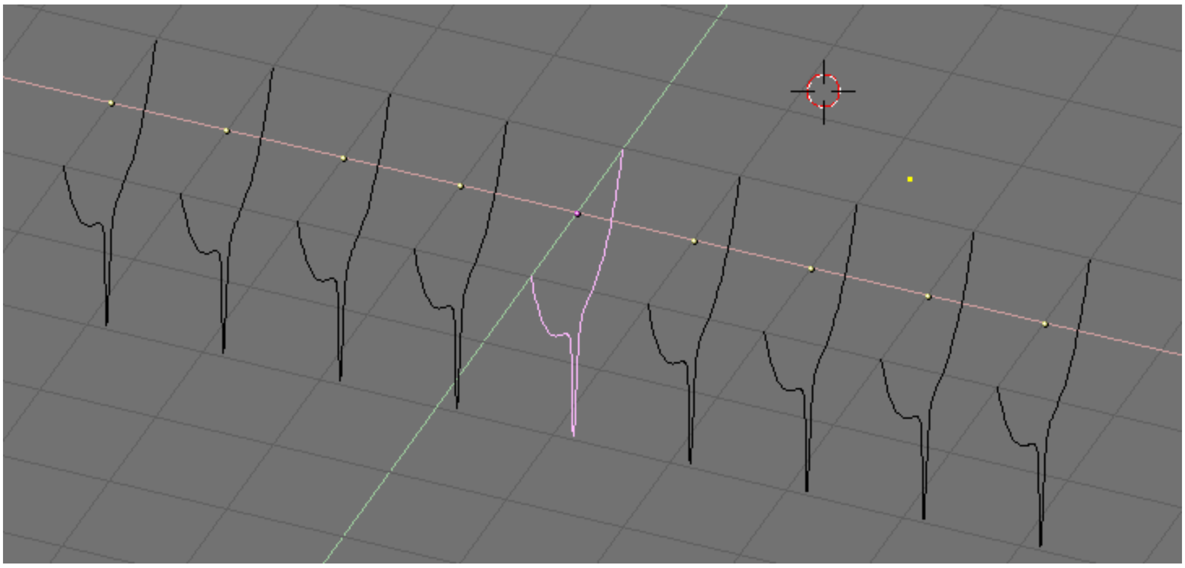
Give the curve the shape of the middle cross section of the boat, by adding vertices as needed with the Split button and, possibly, by setting the NURBS to Endpoint both on 'U' and 'V' (*Profile of the ship.*) as needed.



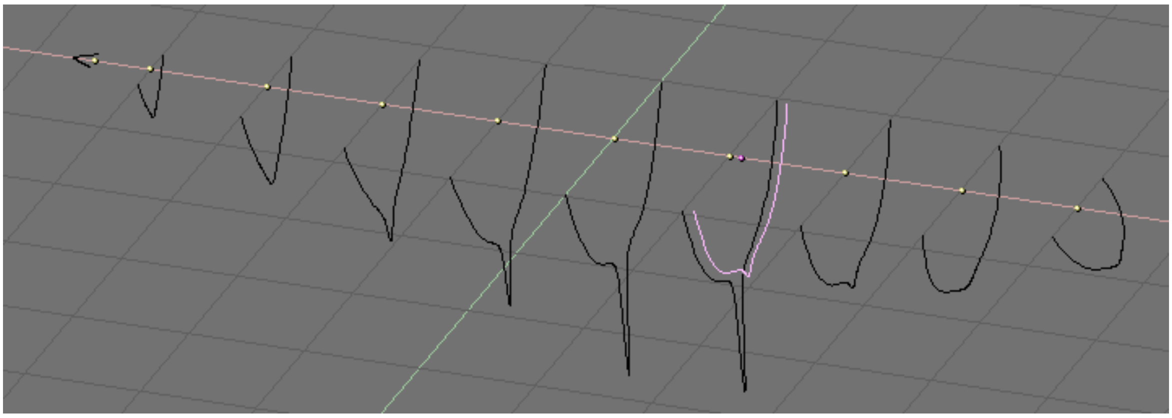
Profile of the ship.

Now duplicate (SHIFT−D) the curve as many times as necessary, to the left and to the right (*Multiple profiles along ship's axis.*). Adjust the curves to match the various sections of the ship at different points along its length. To this end, blueprints help a lot. You can load a blueprint on the background (as we did for the logo design in this chapter) to prepare all the cross section profiles (*Multiple profiles of the correct shapes.*).

Note that the surface we'll produce will transition smoothly from one profile to the next. To create abrupt changes you would need to place profiles quite close to each other, as is the case for the profile selected in *Multiple profiles of the correct shapes.*.

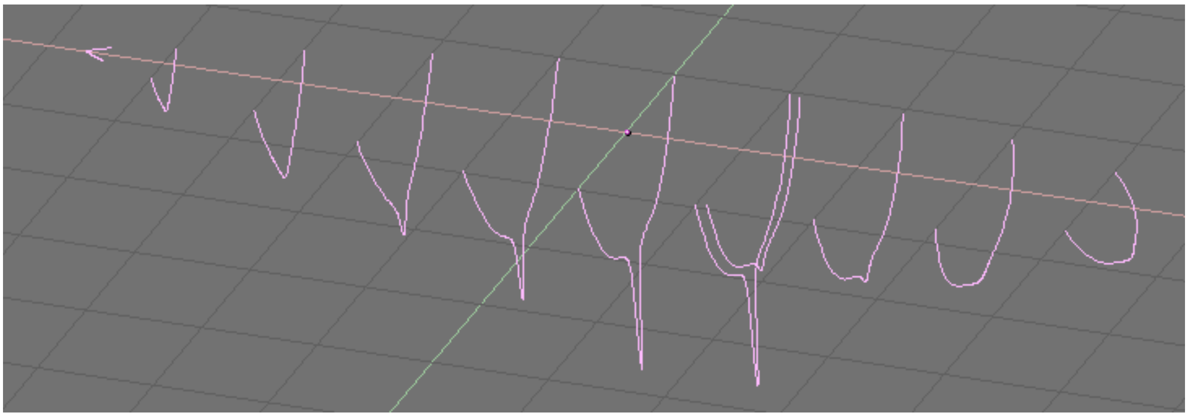


Multiple profiles along ship's axis.



Multiple profiles of the correct shapes.

Now select all curves (with A or B), and join them by pressing CTRL−J and by answering Yes to the question 'Join selected NURBS?'. The profiles are all highlighted in *Joined profiles.*.

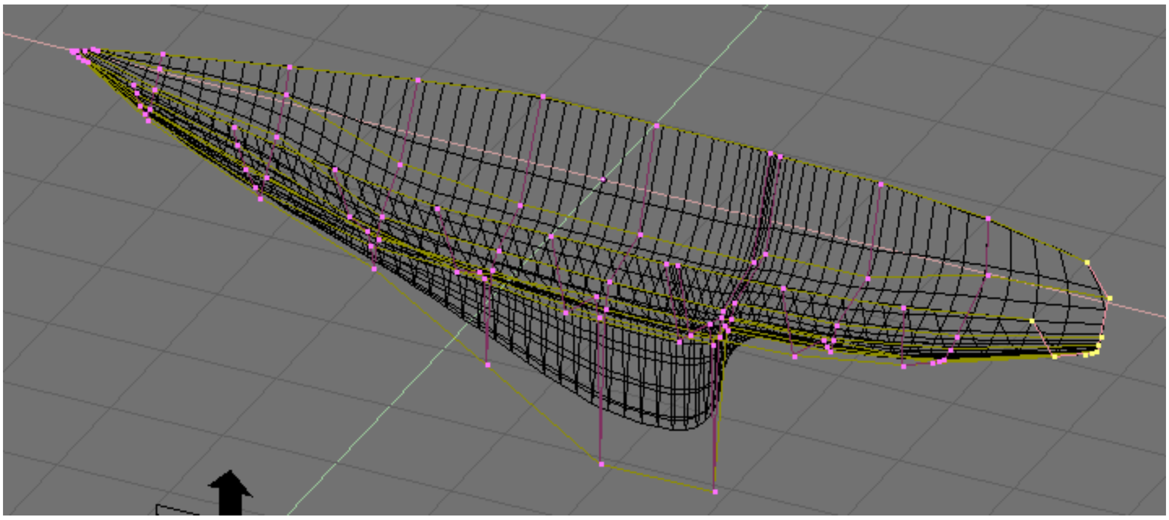


Joined profiles.

Now switch to EditMode (TAB) and select all control points with A; then press F. The profiles should be 'skinned' and converted to a surface (*Skinned surface in edit mode.*).

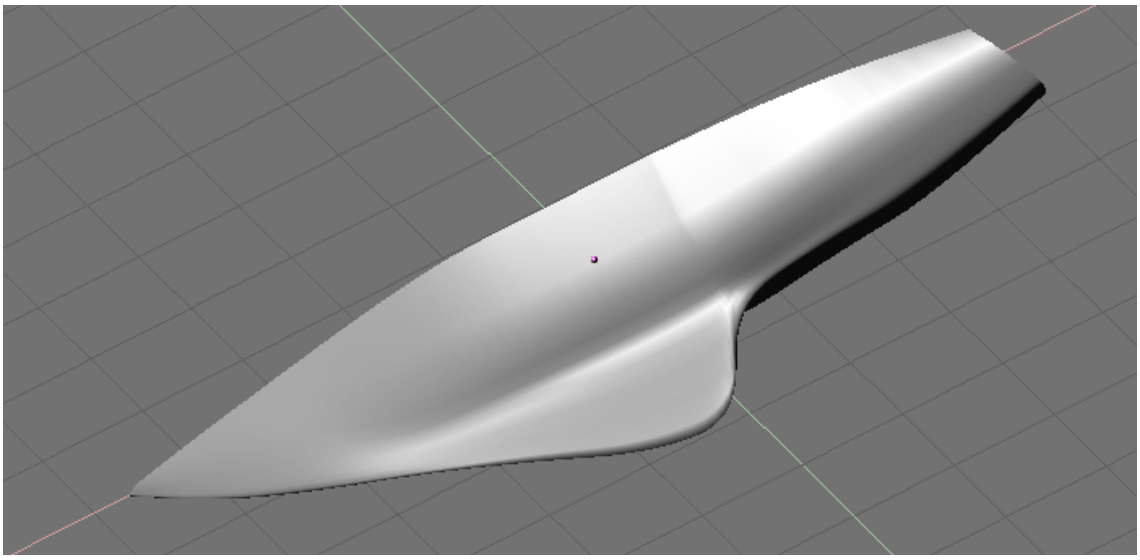
Note

As should be evident from the first and last profiles in this example, the cross−sections need not be defined on a family of mutually orthogonal planes.



Skinned surface in edit mode.

Tweak the surface, if necessary, by moving the control points. *Final hull.* shows a shaded view. You will very probably need to increase ResolU and RelolV to obtain a better shape.



Final hull.

Profile setup

The only limitation to this otherwise very powerful technique is that all profiles must exhibit the same number of control points. This is why it is a good idea to model the most complex cross section first and then duplicate it, moving control points as needed, without adding or removing them, as we've shown in this example.

Previous: Manual/Editing Surfaces | Contents | *Next:* Manual/Text

User Manual: Contents | Guidelines | Blender Version **2.40**

Text

Mode: Edit Mode (Text)

Panel: Editing Context Text

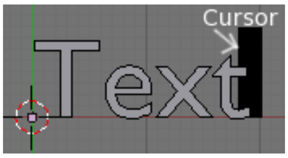
Hotkey: F9

Menu: Add Text

Description

Text is considered a special Curve type that is completely seperate from any of the other Curve types. Not only does the **Font system** have its own *built−in* font but it can use external fonts too, including *PostScript Type 1*, *OpenType* and *TrueType* fonts.

Options



Created text

Creating a Text object is simple, use Add â†’ Text. Once the text is created you are placed in Edit mode with the word "**Text**" inserted as a default placeholder, see (*Created text*). The *Black* block is the cursor.

Examples

Text Examples.

(*Text Examples*) shows some examples of various fonts in action including the "blue" font that has been applied to a curve path.

Hints

A maximum of **50000** characters is allowed per text object, however, be forewarned that the more characters a single text object has, the slower the object will respond interactively.

Editing Text

Mode: Edit Mode (Text)

Hotkey: see below

Description

Editing Text is similar to using a standard text editor but is not as full featured and is slightly different.

Options

Exit Edit Mode
: Tab doesn't insert a tab character in the text, but rather enters and exits edit mode, as with other object types.

Copy
: To copy text to the buffer use Ctrl C.

Cut and Copy
: To cut and copy text to the buffer use Ctrl X.

Paste
: To paste text from the buffer use Ctrl V.

Delete all text
: To completely erase or delete all text Ctrl Backspace.

Home/End;
: Homeand End move the cursor to the begining and end of a line respectively.

Next/Previous word
: To move the cursor on word a boundry use Ctrl â† or Ctrl â†’.

The text buffer does not communicate with the desktop. It only works from within Blender. To insert text from outside Blender see Inserting text.

Selecting text consists of holding down the Shift while using the **Arrow** â†’ keys or Page Up / Page Down keys. The selection is remembered even in Object mode.

Inserting Text

You can insert text three different ways: from the internal text buffer (Editing), the "Lorem" button (Misc) or a text file.

To load text from a text file click the "**Insert Text**" button on the Font panel. The will bring up a "File Browser" window for navigating to a valid UTF−8 file. As usual, be careful that the file doesn't have too many characters as interactive response will slow down.

Changing Fonts

Mode: Edit Mode (Text)

Panel: Editing Context Text

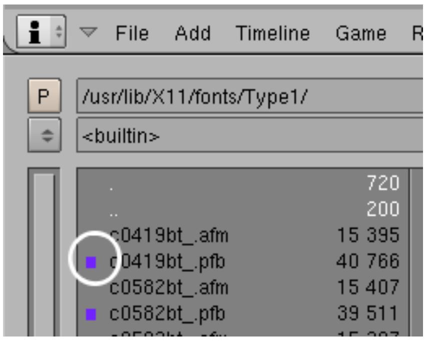
Hotkey: F9

Description

Blender comes with a *built−in* font by default and is displayed in the listbox next to the "Load" button on the Font panel. The *built−in* font is always present and shows in the listbox as "**<builtin>**".

Font listbox and button

Options



Loading a Type 1 font file

To use a different Font you need to load it first by clicking the "Load" button in the Font panel and navigating to a *valid* font. The "File Browser" window will highlight any valid fonts by placing a small purplish rectangle next to each valid entry as shown in (*Loading a Type 1 font file*). The white circle highlights an example of a valid font.

Un*x note

Fonts are typically located under /usr/lib/fonts, or some variant like /usr/lib/X11/fonts, but not always. They may be in other locations as well, such as /usr/share/local or /usr/local/share, and possibly related subtrees.

If you select a font that Blender can't understand, you will get the error "**Not a valid font**".

A seperate font is required for each style. For example, you will need to load an *Italics* font in order to make characters or words italic. Once the font is loaded you can apply that font "Style" to the selected characters or the whole object. In all, you would need to load a minimum of three different types of fonts to represent each style (**Normal**, **Italics**, **Bold**).

Typography

Mode: Edit Mode (Text)

Panel: Editing Context Text

Hotkey: F9

Description

Blender has a number of typographic controls for changing the style and layout of text.

Options

Bold, Italics and Underline

Italics
: Toggled with Ctrl I, font set with the "I" button.

Bold
: Toggled with Ctrl B, font set with the "B" button.

Underline
: Toggled with Ctrl U or by using the "U" button.

Blender's B and I buttons don't work the same way as other applications. They serve as placeholders for you to load up certain fonts manually, which get applied when you use Ctrl B or Ctrl I when editing text.

To apply the Bold/Italics/Underline attribute to a set of characters you either turn on Bold/Italics/Underline prior to selecting characters or highlight first and then toggle Bold/Italics/Underline with a hotkey. Bold/Italics/Underline is applied based on a loaded font. For example, some characters may have one font representing normal characters and the *builtin* font representing Bold; see (*Bold text*).

Bascially each font style is represented by a loaded font. One font may represents Bold while another font represents Italics (i.e. One font per style.)

Alignment

Flush
: **Always** flushes the line, even when it's still being entered, it uses character spacing (kerning) to fill lines.

Justify
: Only flushes a line when it is **terminated** either by wordwrap or by Enter, it uses *whitespace* instead of *character spacing* (kerning) to fill lines.

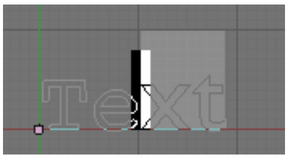
Both "Flush" and "Justify" only work within frames.

Word spacing
: A factor by which whitespace is scaled in width.

Kerning

Manual *kerning*, between any pair of characters, can be controlled by press Alt â† or Alt â†’ to decrease/increase kerning by steps of 0.1.

Examples



Bold text

In (*Bold text*), one font is used for "**Te**" and a different font for "**xt**".

Text Frames

Mode: Object Mode / Edit Mode (Text)

Panel: Editing Context Text

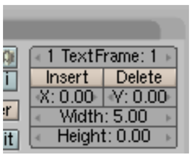
Hotkey: F9

Description

Text Frames allow you to distribute the text amongst rectangular areas within a single text object. An arbitrary number of freely positionable and resizable text Frames are allowed per text object.

Text flows continuously from the lowest−numbered Frame to the highest−numbered Frame with text inside each frame word−wrapped. Text flows between Frames when a lower−numbered frame can't fit anymore text. If the last Frame is reached text overflows out of the Frame.

Options



Text Frame

Frames are controlled from the upper right corner of the Font panel; see (*Text Frame*).

Frame size

By default the first Frame for a new text object, and any additional Frames, has a size of **Zero** for both **Width** and **Height**, which means the Frame is initially not visible.

Frames with a width of **0** are ignored completely during text flow (no wordwrap happens) and Frames with a height of **0** flow forever (no flowing to the next textframe).

In order for the frame to become visible the Frame's width must be greater than **0**.

> Note
>
> Technically the height is never actually **0** because the font itself always contributes height.

Frame width

(*Frame width*) is a text object with a width of **5.0**. And because the frame width is greater than **0** it is now visible and is drawn in the active theme colour as a dashed rectangle. The text has overflowed because the text has reached the end of the last frame, the default frame.

Adding−Deleting a Frame

To add a Frame click the "Insert" button on the Font panel. A new frame is added with a default width and height of **0** which means it is not visible, nor will text flow from it into another frame. Be sure to set an offset for the new frame in the X and Y fields. Just an X setting will create a new column.

To delete a Frame click the "Delete" button on the Font panel. Any text in higher frames will be re−flowed downward into lower frames.

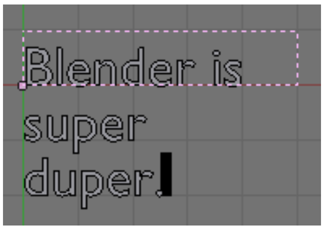
Examples

Text Frames are very similar to the concept of *frames* from a desktop publishing application. You use frames to control the placement and flow of text.

Text Flow

Text 1

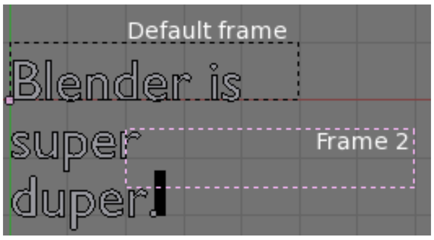
With two or more frames you can organize text to a finer degree. For example, create a text object and enter "**Blender is super duper**"; see (*Text 1*). This text object has a frame, it just isn't visible because the width is **0**.



Text 2

Set the width to **5.0**. The frame is now visible and text is wrapping according to the new width, as shown in (*Text 2*). Notice that the text has overflowed out of the frame. This is because the text has reached the end of the last frame which just happens to be the default/initial frame.

When we add another frame and set its width and height the text will flow into the new frame.

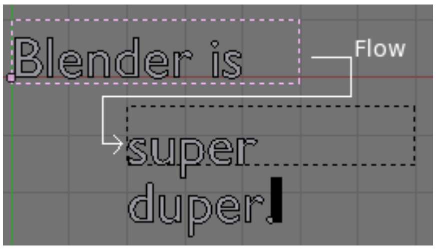


Text 3

Clicking on "Insert" will add a new frame (labeled "Frame 2" in (*Text 3*)) with the same attributes as the previous frame (labeled "Default frame" in (*Text 3*)).

Notice that the text has not yet flowed into this new frame. That is because the previous, or lower numbered, frame has a height of **0**.

Remember the height field may be **0** but the font itself contibutes height. The font's height does not count. This means the height field value is an addition to the font's height.

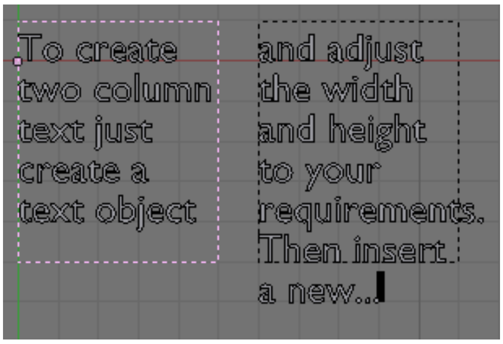


Text 4

To get text to flow *into* "Frame 2" we need to change the height of the default/initial frame. In (*Text 4*) the height of the initial frame −−in pink−− has been increased to **0.1**.

Now the text flows from the initial frame into "Frame 2". Notice that the text overflows out of "Frame 2". Again this is because the text has reached the end of the last frame.

Multiple columns



Text 5

To create two columns of text just create a text object and adjust the initial frame's width and height to your requirements, then insert a new frame. The new frame will have the same size as the initial frame. Set the X position to something greater or less than the width of the initial frame; see (*Text 5*).

Multiple Materials

Mode: Object Mode / Edit Mode (Text)

Panel: Editing Context Link and Materials

Hotkey: F9

Description

Each character can have a different Material index in order to have different materials on different characters.

Options

You can assign indices either as you type, or after by selecting blocks of text and clicking on the "Assign" button in the Link and Materials Panel.

Examples

Red Green Blue

For example to create (*Red Green Blue*) you would need to create three seperate Materials and three seperate Material indexs. Each word would be assigned a Material index by selecting the characters for each word and clicking the "Assign" button. (*Red Green Blue*) is still one single Text object.

Curve and Surface attributes

Mode: Object Mode / Edit Mode (Text)

Panel: Editing Context Curve and Surface

Hotkey: F9

Description

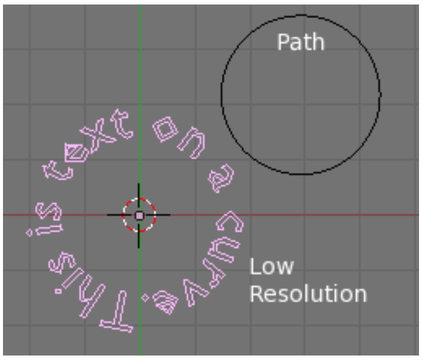
Text is very similar to 2D curves in that they have curve like properties. For example, you can change the Resolution from the Curve and Surface panel to have smooth or coarse text. Once the text is created you can extrude, bevel or even change the thickness.

Options

Because a Text object is similar to a curve it can be converted into a curve using Alt C. Once this is performed the Text becomes a curve and can be manipulated just like a curve. This allows complete control over the shape of the characters beyond what a Text object provides.

The transform from Text to Curve is not reversible; consider saving prior to converting. Also, you can continue to convert the Curve into a Mesh for even further control.

Examples



Rough text

In (*Rough text*) the Resolution has been set to the lowest setting to produce very blocky text. Almost as if the text was broken out of a rock mine. In addition, the text has been applied to a 2D Bezier circle curve.

The path the text has been applied to is labeled "**Path**". To specify a *Curve* or *Path* enter the name of a 2D curve in the "TextOnCurve" field in the Font panel as shown in (*TextOnCurve*). In this example the "**Path**"'s name is "CurveCircle".

See Also

- Modelling: Curves
- Modelling: Surfaces.

Special Characters

Mode: Edit Mode (Text)

Menu: Text Special Characters

Description

There are a few special characters that are available using the **Alt** key or the "Text" menu on the 3D window header. These "Key" combinations are only available while in Edit mode.

Options

A summary of these characters follows; just remember you can access these characters from the Char panel as well:

- Alt c: copyright
- Alt f: Currency sign
- Alt g: degrees
- Alt l: British Pound
- Alt r: Registered trademark
- Alt s: German S
- Alt x: Multiply symbol
- Alt y: Japanese Yen
- Alt 1: a small 1
- Alt 2: a small 2
- Alt 3: a small 3
- Alt ?: Spanish question mark
- Alt !: Spanish exclamation mark
- Alt >: a double >>
- Alt <: a double <<

All the characters on your keyboard should work, including stressed vowels and so on. If you need special characters (such as accented letters, which are not there on a US keyboard) you can produce many of them using a combination of two other characters. To do so, press Alt Backspace within the desired combination, and then press the desired combination to produce the special character. Some examples are given below.

- A, Alt Backspace, ~: ã
- A, Alt Backspace, ,: à
- A, Alt Backspace, `: á
- A, Alt Backspace, O: å
- A, Alt Backspace, ": ë
- A, Alt Backspace, /: ø

Unicode Characters

Mode: Edit Mode (Text)

Panel: Editing Context Char

Hotkey: F9

Description

The font system understands both ASCII and UNICODE character sets with a panel dedicated to assisting in the selection of extended characters.

Since Blender does not support Unicode text input via the keyboard, not all characters are easily accessable from the keyboard. For those difficult characters the Char panel is provided. This panel simply exposes the entire Unicode character set. The character set can be quite large so paging buttons are provided, "**U**" and "**D**".

When you find the character you looking for just click on it in the grid.

Technical Details

For optimum resource usage only characters that are being used consume memory rather than the entire character set.

Previous: Manual/Surfaces Skinning Contents *Next:* Manual/Meta Objects

User Manual: Contents | Guidelines | Blender Version **2.40**

Meta Objects

Mode: Object Mode or Edit Mode (Meta)

Hotkey: Shift A

Menu: Add Meta

Meta Objects are *implicit surfaces* meaning that they are *not explicitly* defined by vertices (as meshes are) or control points (as surfaces are); they exist *procedurally* (i.e. computed dynamically).

Another way of describing Meta Objects are as fluid *Mercurial*, or *Clay−like* forms that have a "rounded" shape.

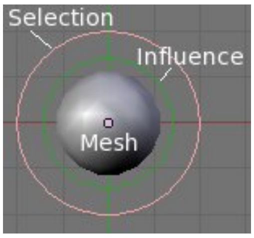
There are five predefined Meta Object configurations:

Ball
    A point underlying structure.
Tube
    A line segment underlying structure.
Plane
    A planar underlying structure.
Elipsoid
    A spherical underlying structure.
Cube
    A volumetric cubic underlying structure.

Each is a different mathematical definition and at any time you can switch between them using the MetaBall tools panel. Each one has an underlying mathematical structure that defines it.

Typically Meta Objects are used for special effects or as a basis for modelling. For example, you could use a collection of Meta Objects to form the initial shape of your model and then convert it another object type for further modelling.

Options



MetaBall example

Each Meta Object always appears with two rings or circles; see (*MetaBall example*).

Selection

The outer ring (labelled "**Selection**" and coloured pink) is for *selecting* and exists because there are two types of elements that can be selected within a Meta Object. You can select the Meta Object itself, by clicking the ring, or select the Mesh by clicking the mesh. Without the selection ring you couldn't select just the Meta Object.

S with the outer ring selected scales the Meta Object

Stiffness

The inner ring (labelled "**Influence**" and coloured green) defines the Meta Object's stiffness value − how much influence it has on other Meta Objects. When a Meta Object comes within "range" of another Meta Object the two Meta Objects will begin to interact with each other. They don't necessarily need to intersect and depending on the Threshold and Stiffness settings they most likely won't need too.

S with the inner ring selected increases or decreases the Stiffness

Technical Details

A more formal definition of a meta object can be given as a *directing structure* which can be seen as the source of a static field. The field can be either positive or negative and hence the field generated by neighbouring directing structures can attract or repel.

The implicit surface is defined as the surface where the 3D field generated by all the directing structures assume a given value. For example a Meta Ball, whose directing structure is a point, generates an isotropic field around it and the surfaces at constant field value are spheres centered at the directing point.

Meta Objects are nothing more than mathematical formulas that perform logical operations on one another (AND, OR), and that can be added and subtracted from each other. This method is also called **Constructive Solid Geometry** (CSG). Because of its mathematical nature, CSG uses little memory, but requires lots of processing power to compute.

Threshold (Influence)

Mode: Object Mode or Edit Mode (Meta)

Panel: Editing Context MetaBall

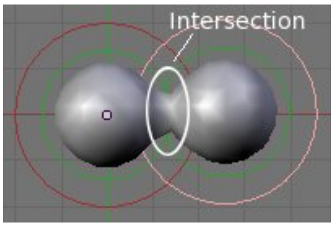
Description

Threshold defines how much a MetaObject's surface "Influences" other MetaObjects. It controls the *field level* at which the surface is computed. The setting is global to a Group of MetaObjects. As the Threshold increases the influence that each MetaObject has on one another increases.

Options

There are two types of influence: **positive** or **negative**. The type can be toggled on the MetaBall tools panel using the "Negative" button. You could think of **positive** as attraction and **negative** as repulsion of meshs. A negative MetaObject will push away or repel the meshes other types of MetaObjects.

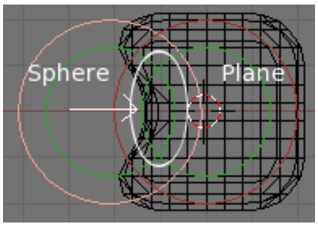
Examples



Positive

A *Positive* influence is defined as an attraction meaning the meshs will stretch towards each other as the *rings of influence* intersect. (*Positive*) shows two MetaBalls' *ring of influence* intersecting with a *positive* influence.

Notice how the meshes have pulled towards one another. The area circled in white shows the green *influence* rings intersecting.



Negative

The opposite effect of a *positive* influence would be a *negative* influence. (*Negative*) shows a MetaBall and MetaPlane where the MetaBall is negative and the MetaPlane is positive. Negative MetaObjects are not visible to indicate that they are configured as such.

The white arrow indicates how the sphere is repelling or pushing away the plane's mesh. This causes the plane's mesh to cave in or collapse inward. If you move the plane away from the sphere the plane's mesh will restore itself.

Wiresize

Mode: Object Mode or Edit Mode (Meta)

Panel: Editing Context MetaBall

Description

The Wiresize controls the resolution of the resultant mesh as generated by the MetaObject.

Options

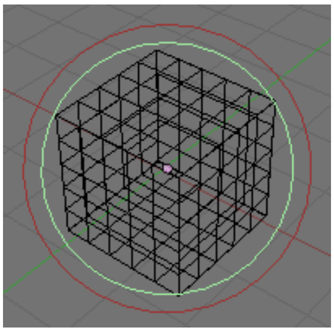
Wiresize
: The 3D View resolution of the generated mesh. The range is from "**0.05**" (finest) to "**1.0**" (coarsest).

Rendersize
: The rendered resolution of the generated mesh. The range is from "**0.05**" (finest) to "**1.0**" (coarsest).

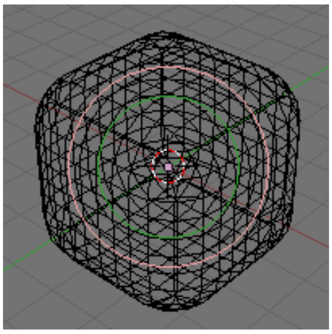
Examples

One way to see the underlying mathematical structure is to lower the Wiresize, increase the Threshold and set the Stiffness a fraction above the Threshold.

(*Underlying structure*) is a (*MetaCube*) with the above mentioned configuration applied as follows: Wiresize of "**0.410**", Threshold of "**5.0**" and Stiffness a fraction above at "**5.01**".



Underlying structure.



MetaCube.

You can clearly see the underlying Cube structure that gives the MetaCube its shape.

Stiffness

Mode: Edit Mode (Meta)

Panel: Editing Context MetaBall Tools

Hotkey: S

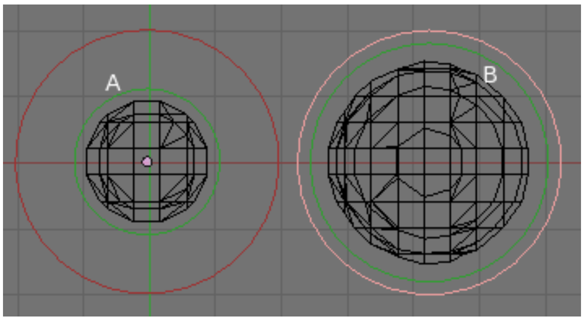
Description

Together with Threshold, Stiffness controls the influencing range. Stiffness directly controls the *Green ring* surrounding a MetaObject. The field is found on the MetaBall tools panel.

Options

The range is from "**0.0**" to "**10.0**". But to be visible the Stiffness must be slightly larger than the Threshold value. You can visually adjust the Stiffness ring by using the RMB to select it and activating Scale mode with the S.

Examples



Stiffness

In (*Stiffness*), the MetaBall labeled "**A**", has a smaller Stiffness value than the MetaBall, labeled "**B**". As you can see the *Green ring* radius is different between them.

Grouping

Mode: [Object Mode] or [Edit Mode] (Meta)

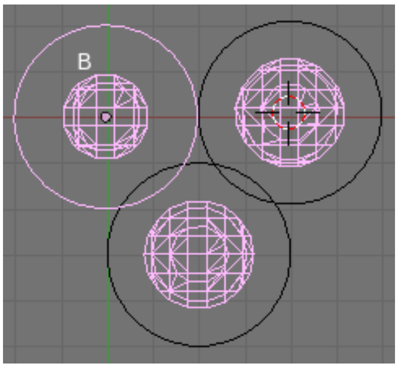
Panel: Editing Context Link and Materials

Hotkey: F9

Description

MetaObjects are grouped by the *Family* part of an Object name; the **OB:** field in most panels, **NOT** the **MB:** field. The Object name is broken into two parts, the left part before the period and the right part after the period. For example, the *Family* part of "MetaPlane.001" is "MetaPlane".

Options



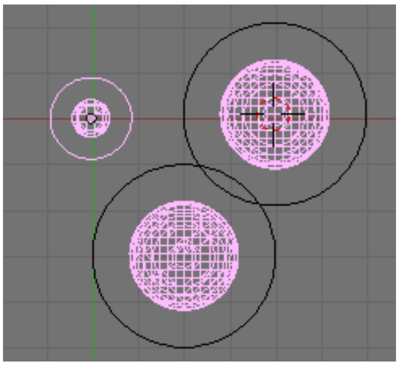
MetaBall Base

Groups of MetaObjects are controlled by a *base* MetaObject which is identified by an Object name **without** a "number" part. For example, if we have five MetaObjects called "MetaThing", "MetaThing.001", "MetaThing.002", "MetaThing.003", "MetaThing.004", the *base* MetaObject would be "MetaThing".

The *base* MetaObject determines the basis, the resolution, *and* the transformations. It also has the Material and texture area. The *base* MetaObject is the parent of the other MetaObjects in the group.

Examples

(*MetaBall Base*) shows the *base* MetaObject labeled "**B**". The other two MetaObjects are *base* children. Children's selection rings are always black while the Group's mesh is pink. Because the MetaObjects are grouped they form a unified mesh which can always be selected by selecting the mesh of any MetaObject in the group. For example, selecting the mesh of the lower sphere in (*MetaBall Base*) would result in the very same appearance that you see **in** (*MetaBall Base*); both the *base* and the children's mesh would be highlighted.



Scaling the "base"

The *base* MetaObject controls the polygonalization (mesh structure) for the group and as such controls the polygonalization for the children (*non−base*) MetaObjects. If we transform the *base* MetaObject the children's polygonalization changes. However, if we transform the children the polygonalization remains unchanged.

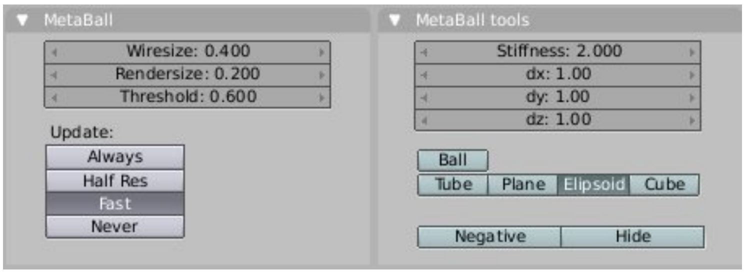
Hints

This doesn't mean the meshes don't deform towards or away from each other. It means the underlying mesh structure changes only when the *base* transforms. For example, if you Scale the *base* the children's mesh structure changes. In (*Scaling the "base"*), the *base* has been scaled down which has the affect of scaling the mesh structure of the children. As you can see the children's mesh resolution has increased while the *base* decreased.

A group can only have one Material and Texture area. This normalises the coordinates of the vertices. Normally the texture area is identical to the *bounding box* of all vertices. The user can force a texture area with the T command in Object mode.

Previous: Manual/Text	Contents	*Next:* Manual/Editing Meta Objects

User Manual: Contents | Guidelines | Blender Version **2.42**



Metaballs use a different modelling workflow from more traditional techniques. You guide their shape rather then editing their vertices. Use the Editing buttons to guide their shape.

The Metaball panel controls how they look in your 3D view, and how smooth they are when rendered. Choose a smaller Rendersize to make them smoother when rendering. For real−time updates while you are working with them inside Blender, choose a larger Wiresize. The Threshold sets how close they can get to each other before merging. The Update: selections choose your display update frequency. Use Fast to save CPU time and increase Blender's responsiveness to you. Use Never if you really want to get confused between your display and what is rendered.

The Metaball tools panel controls their shape. Select Ball for a sphere, Tube, Plane, etc. Each choice is like the mesh equivalent with lots of subsurfacing.

Previous: Manual/Meta Objects	Contents	*Next:* Manual/DupliVerts

User Manual: Contents | Guidelines | Blender Version **2.43**

DupliVerts

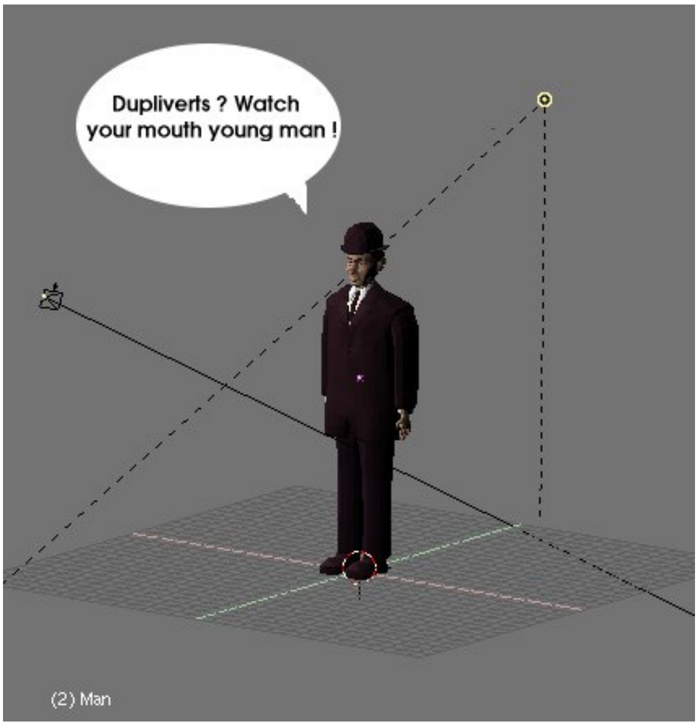
Mode: Object mode

Panel: Anim Settings â†’ DupliVerts

Hotkey: F7

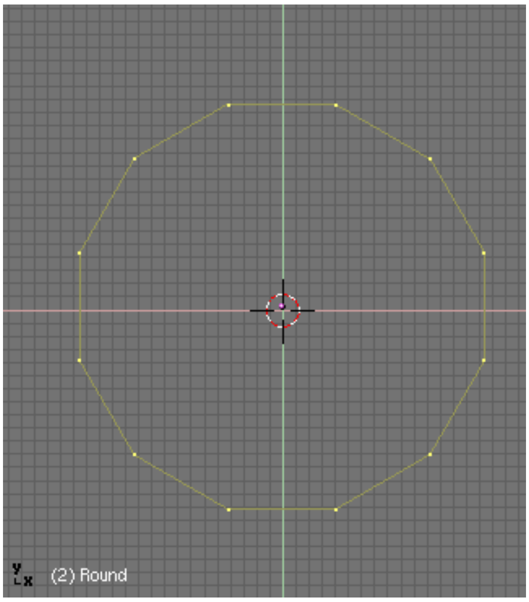
DupliVerts are not a rock band nor a dutch word for something illegal (well maybe it is) but is a contraction for Duplication at Vertices, meaning the duplication of a base Object at the location of the Vertices of a Mesh (or even a Particle system). In other words, when using DupliVerts on a mesh, an instance of the base object is placed on every vertex of the mesh. There are actually two approaches to modelling using DupliVerts. They can be used as an arranging tool, allowing us to model geometrical arrangements of objects (e.g. the columns of a Greek temple, the trees in a garden, an army of robot soldiers, the desks in a classroom). The object can be of any object type which Blender supports. The second approach is to use them to model an Object starting from a single part of it (i.e.: the spikes in a club, the thorns of a sea−urchin, the tiles in a wall, the petals in a flower).

DupliVerts as an Arranging Tool



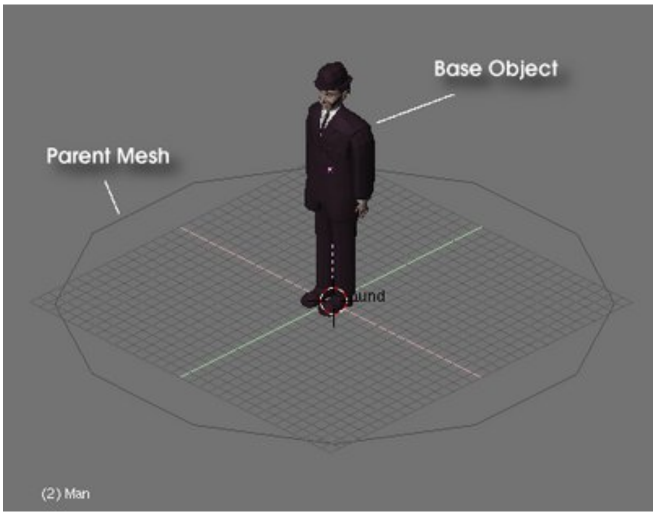
A simple scene to play with.

All you need is a base object (e.g. the *tree* or the *column*) and a pattern mesh with it's vertices following the pattern you have in mind. In this section, we will use a simple scene for the following part. It consists of a camera, the lamps, a plane (for the floor) and a strange man I modelled after Magritte's famous character (*A simple scene to play with.*). If you don't like surrealism you will find this part extremely boring.



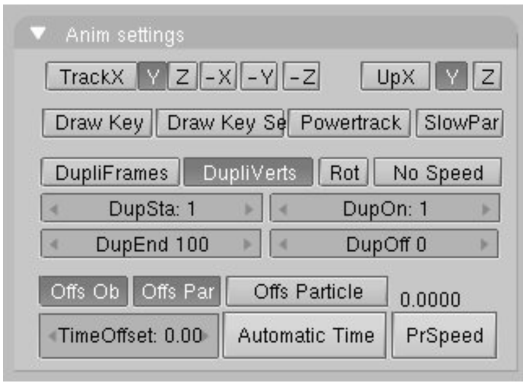
A circle for a parent mesh

Anyway, the man will be my *base Object*. It is a good idea that he will be at the center of the co−ordinate system, and with all rotations cleared. Move the cursor to the base object's center, and from Top View add a mesh (the example uses a circle as a pattern, with 12 vertices or so (*A circle for a parent mesh*). The pattern object can be a two−dimensional primitive (plane or circle), or even a three−dimensional primitive mesh (cube, tube, sphere) or a curve (two dimensional, or a three−dimensional path), or even your own custom mesh, so long as it has vertices (you cannot use a camera, for example, but you can use a large landscape mesh if you want to plant trees − trees being your base object).



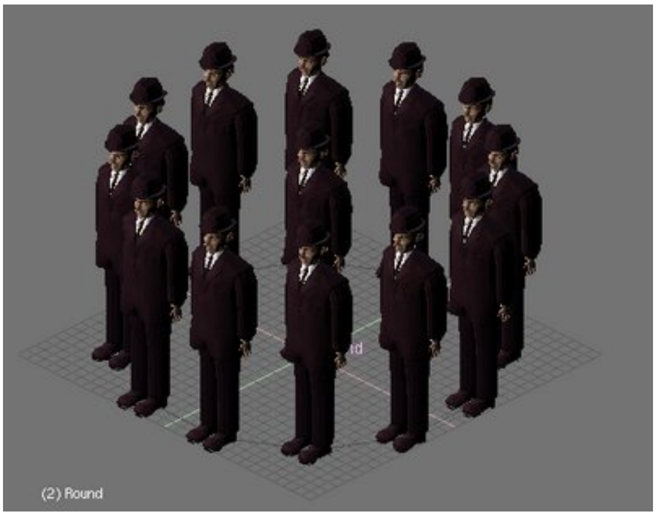
The man is parented to the circle.

Out of Edit Mode, select the base Object and add the circle to the selection (order is very important here). Parent the base object to the circle by pressing Ctrl P. Now, the circle is the parent of the character (*The man is parented to the circle.*).



The Animation Buttons

Now select only the circle, switch the Buttons Window to the Object Context (via or F7) and select the DupliVerts Button in the Anim Settings Panel (*The Animation Buttons*).

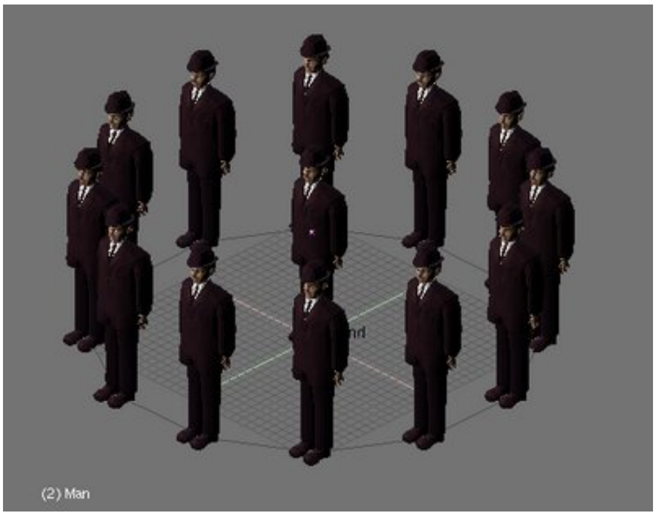


In every vertex of the circle a man is placed.

Wow, isn't it great? Don't worry about the object at the center (*In every vertex of the circle a man is placed.*). It is still shown in the 3D−views, but it will *not* be rendered. You can now select the base object, change (scale, rotate, Edit Mode) (and also Object Mode, however scaling in Object Mode could bring up some problems when applying Rotation to DupliVerts as we will see soon) it and all DupliVerted objects will reflect the changes. But the more interesting thing to note is that you can also edit the parent circle.

Note

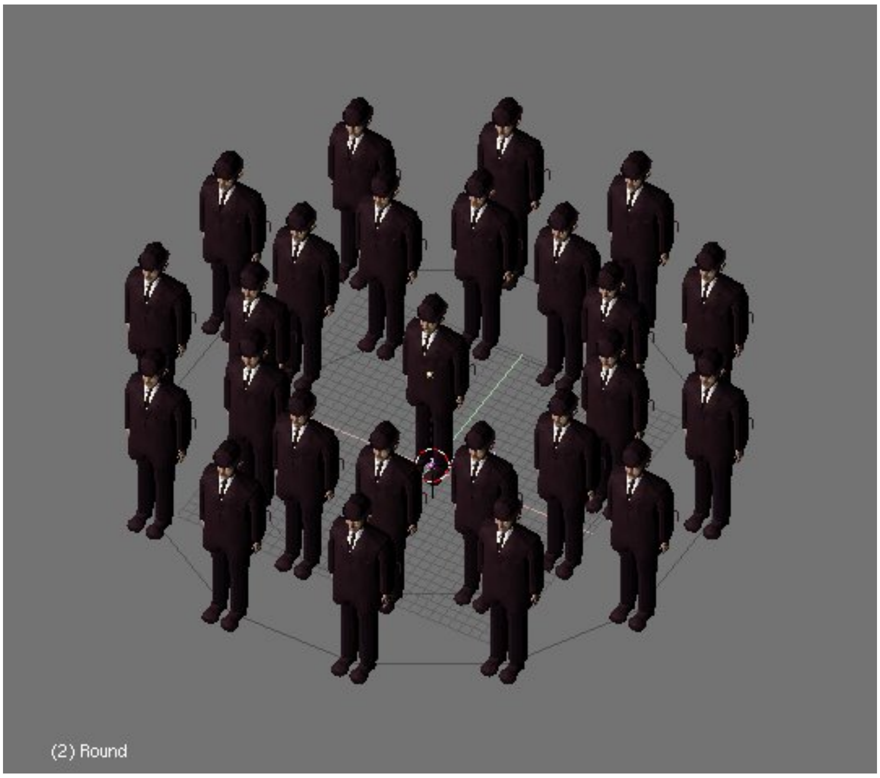
The base Object is not rendered if DupliVerted on a Mesh but it *is* rendered if DupliVerted on a Particle System! This doesn't appear to be true on Blender 2.45 and later.



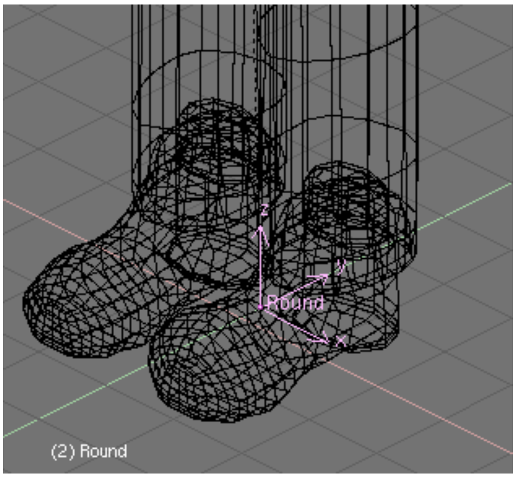
Changing the size of the circle in Edit Mode.

Select the circle and scale it. You can see that the mysterious men are uniformly scaled with it. Now enter the Edit Mode (Tab) for the circle, select all vertices A and scale it up about three times. Leave Edit Mode and the DupliVerted objects will update (*Changing the size of the circle in Edit Mode.*). This time they will still have their original size but the distance between them will have changed. Not only can we scale in Edit Mode, but we can also delete or add vertices to change the arrangement of men.

Select all vertices in Edit Mode and duplicate them (Shift D). Now scale the new vertices outwards to get a second circle around the original. Leave Edit Mode, and a second circle of men will appear (*A second row of Magritte's men.*).



A second row of Magritte's men.

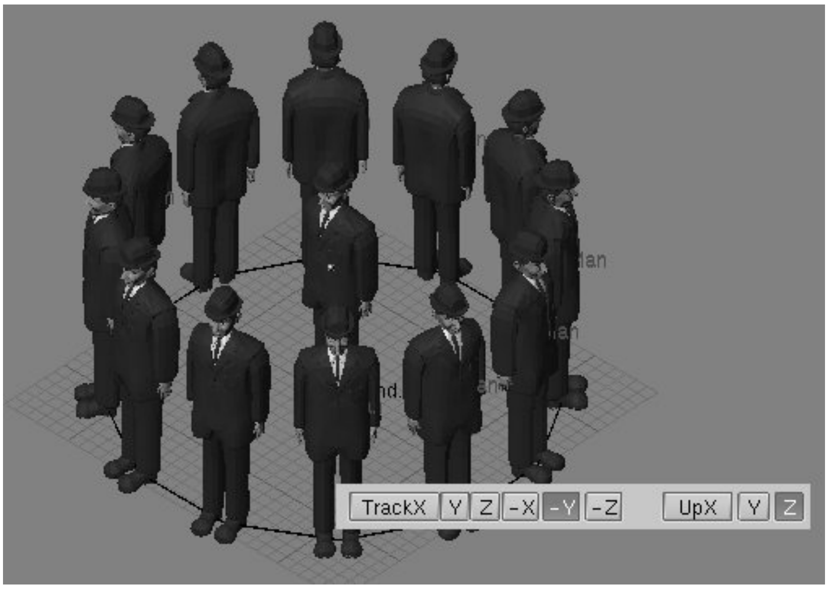


Show object's axis to get what you want.

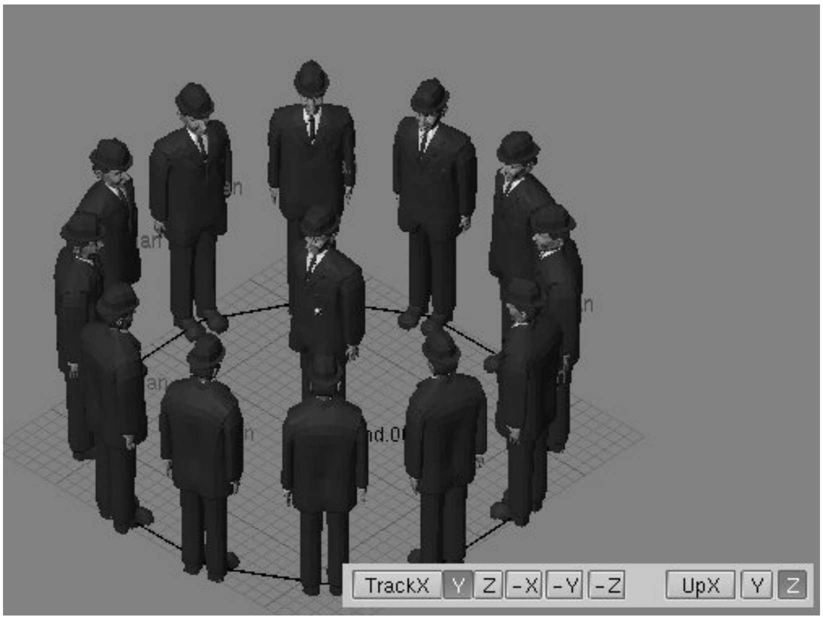
Until now all Magritte's men were facing the camera, ignoring each other. We can get more interesting results using the Rot Button next to the DupliVerts button in the Anim Settings Panel. With this Toggle−Button active, we can rotate the DupliVerted objects according to the normals of the parent Object. More precisely, the DupliVerted Objects axis are aligned with the normal at the vertex location. Which axis is aligned (X, Y or Z) with the parent mesh normal depends on what is indicated in the TrackX, Y, Z buttons and the UpX, Y, Z buttons top in the Anim Settings Panel. Trying this with our surrealist buddies, will lead to weird results depending on these settings.

The best way to figure out what will happen is first of all aligning the "base" and "parent" objects' axis with the World axis. This is done selecting both objects and pressing Ctrl A, and click the Apply Size/Rot? menu.

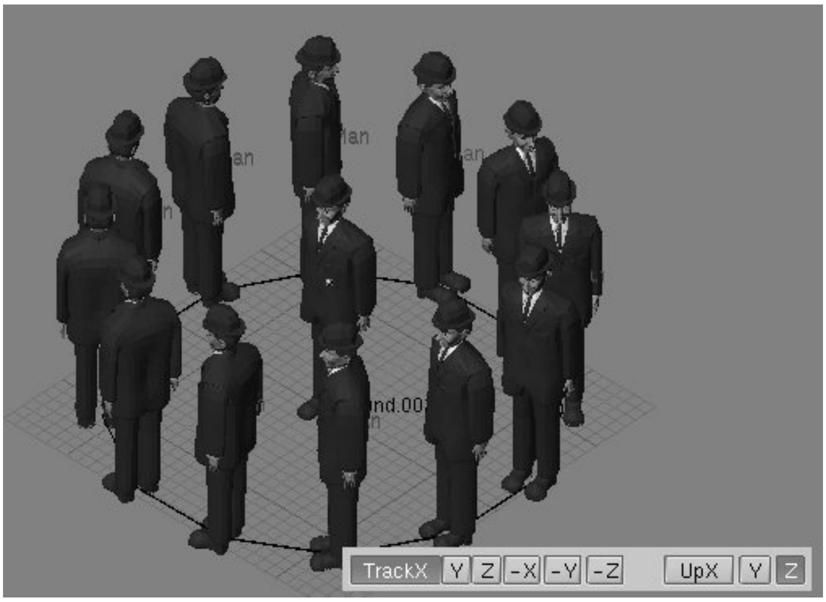
Then make the axis of the base object and the axis and normals in the parent object visible (*Show object's axis to get what you want.* − in this case, being a circle with no faces, a face must be defined first for the normal to be visible − actually to exist at all). Now select the base object (our Magritte's man) and play a little with the Tracking buttons. Note the different alignment of the axis with the different combinations of UpX, Y, Z and TrackX, Y, Z (*Negative Y Axis is aligned to vertex normal (pointing to the circle's center)*, *Positive Y axis is aligned to normal*, *Positive X axis is aligned to normal*, *Positive Z axis is aligned to normal (weird, huh?)*).



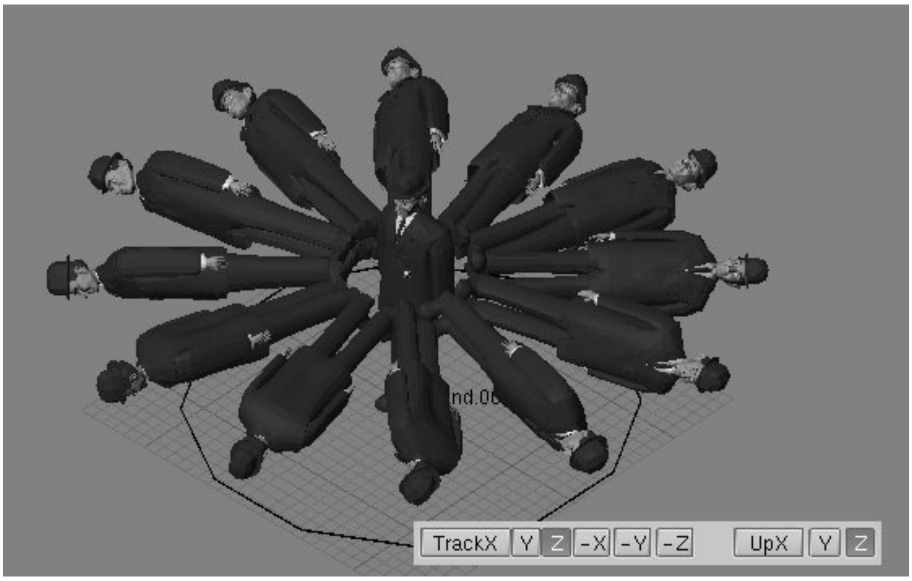
Negative Y Axis is aligned to vertex normal (pointing to the circle's center)



Positive Y axis is aligned to normal



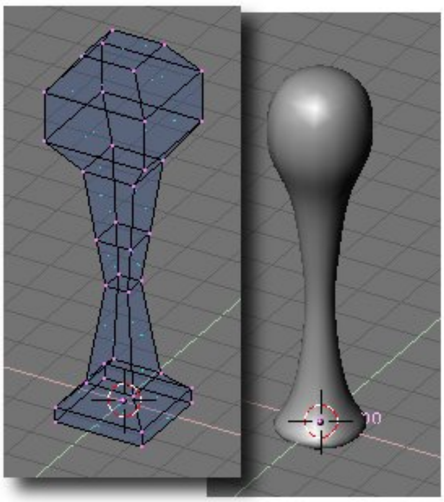
Positive X axis is aligned to normal



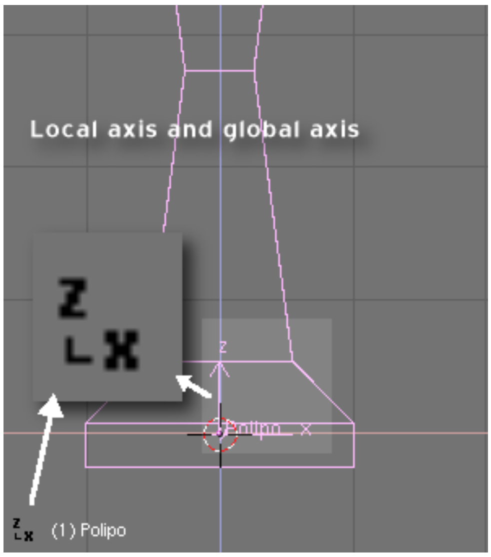
Positive Z axis is aligned to normal (weird, huh?)

DupliVerts to Model a Single Object

Very interesting models can be made using DupliVerts and a standard primitive. Starting from a cube in Front View, and extruding a couple of times I have modelled something which looks like a tentacle when SubSurfs are activated (*Strange tentacle and SubSurfed version.*). Then I added an Icosphere with 2 subdivisions.



Strange tentacle and SubSurfed version.

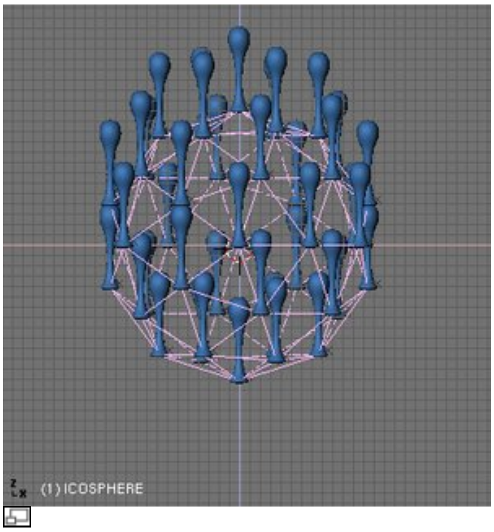


Local reference of the tentacle.

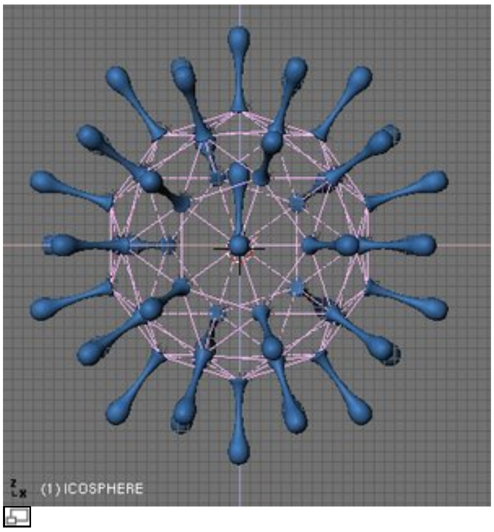
I had to take special care to be sure that the tentacle was located at the sphere center, and that both the tentacle axis and the sphere axis were aligned with the world axis as above (*Local reference of the tentacle.*).

Now, I simply make the icosphere the parent of the tentacle. Select the icosphere alone and made it DupliVert in the Anim Settings Panel (*DupliVerts not rotated.*).

Press the Rot button to rotate the tentacles (*DupliVerts rotated.*).

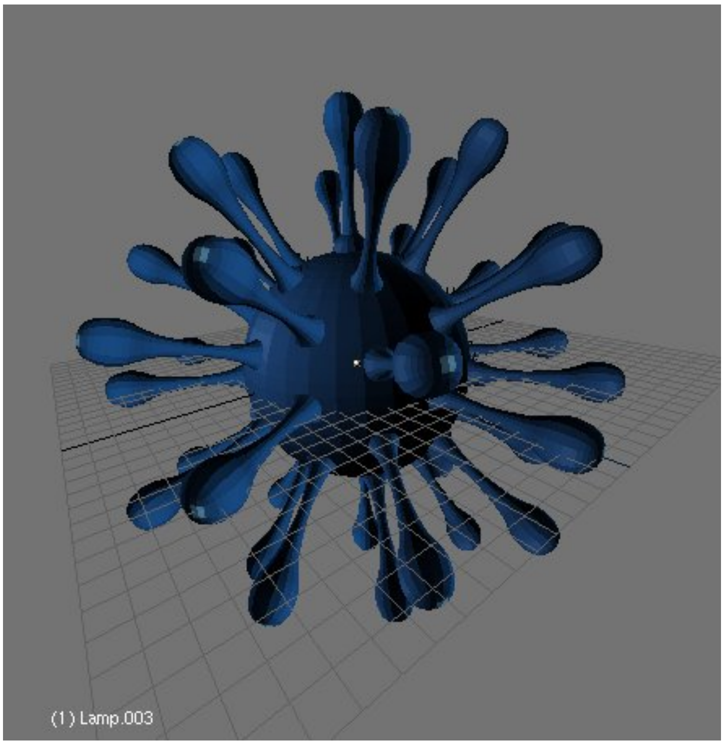


DupliVerts not rotated.



DupliVerts rotated.

Once again to make the tentacle point outwards we have to take a closer look to it's axis. When applying Rot, Blender will try to align one of the tentacle's axis with the normal vector at the parent mesh vertex. We didn't care about the Parent circle for Magritte's men, but here we should care about the Sphere, and you will soon notice that it is *not* rendered. You probably would like to add an extra renderable sphere to complete the model. You can experiment in Edit Mode with the tentacle, moving it's vertices off the center of the sphere, but the object's center should always be at the sphere's center in order to get a symmetrical figure. However take care not to scale up or down in one axis in Object Mode since it would lead to unpredictable results in the DupliVerted objects when applying the Rot button.



Our model complete.

Once you're done with the model and you are happy with the results, you can select the tentacle and press Shift Ctrl A and click on the Make duplis real ? menu to turn your virtual copies into real meshes (*Our model complete.*).

See also

Other duplication methods are listed here: Manual/Duplication

Previous: Manual/Editing Meta Objects	Contents	*Next:* Manual/DupliFrames

User Manual: Contents | Guidelines | Blender Version **2.31**

DupliFrames

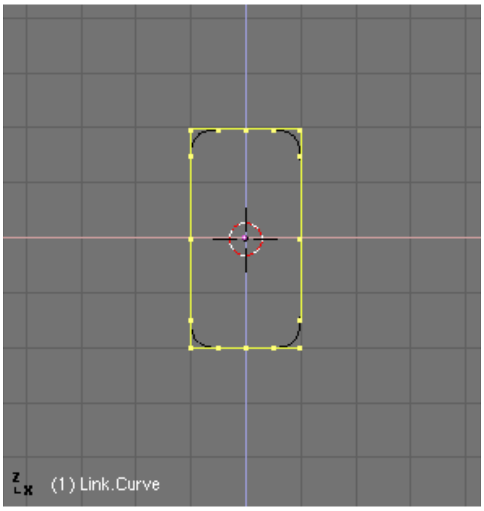
You can consider DupliFrames in two different ways: an arranging or a modelling tool. In a way, DupliFrames are quite similar to DupliVerts. The only difference is that with DupliFrames we arrange our objects by making them follow a curve rather than using the vertex of a mesh. DupliFrames stands for DUPLIcation at FRAMES and is a very useful modelling technique for objects which are repeated along a path, such as the wooden sleepers in a railroad, the boards in a fence or the links in a chain, but also for modelling complex curve objects like corkscrews, seashells and spirals.

Modelling using DupliFrames

We are going to model a chain with it's links using DupliFrames. First things come first. To explain the use of DupliFrames as a modelling technique, we will start by modelling a single link.

To do this, add in front view a Curve Circle (Bézier or NURBS, whatever).

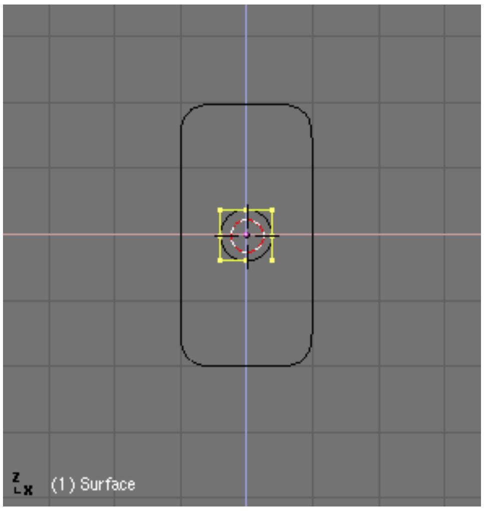
In Edit Mode, subdivide it once and move the vertices a little to fit the link's outline (*Link's outline*).



Link's outline

Leave Edit Mode and add a Surface Circle object (*Link's cross section*).

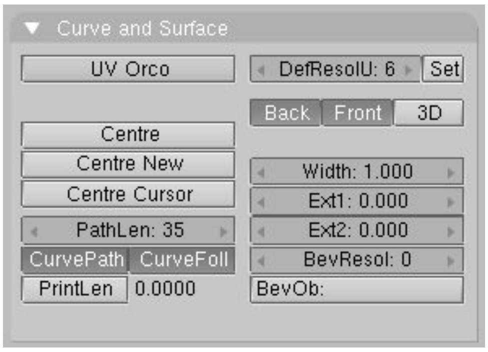
NURBS−surfaces are ideal for this purpose, because we can change the resolution easily after creation, and if we need to, we can convert them to a mesh object. It is very important that you do not confuse Curve Circle and Surface Circle. The first one will act as the shape of the link but it will not let us do the skinning step later on. The second one will act as a cross section of our skinning.



Link's cross section

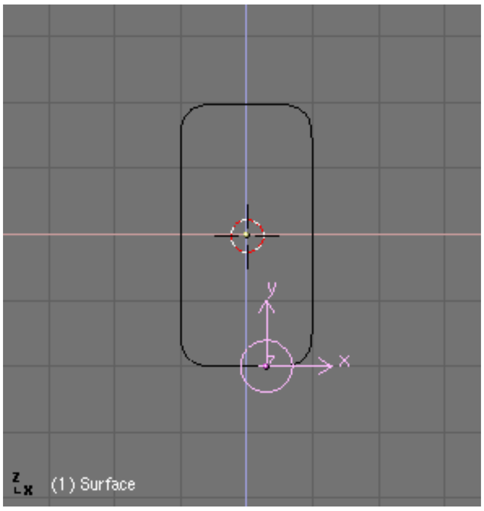
Now parent the circle surface to the circle curve (the link's outline) as a Normal parent (not a Curve Follow constraint).

Select the curve and in the Object Context and Anim Settings Panel press CurvePath and CurveFollow (*Curve's settings: Curve Path and Curve Follow*.).



Curve's settings: Curve Path and Curve Follow.

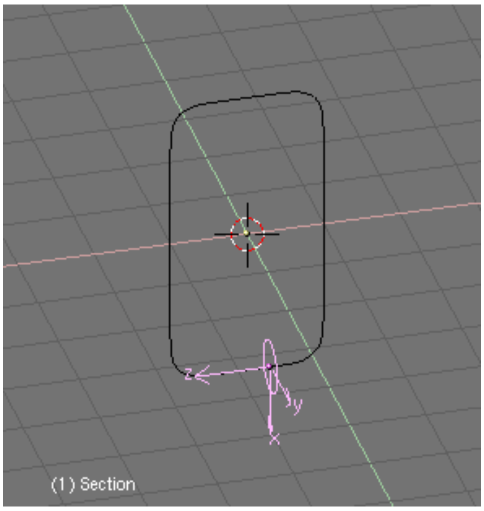
It's probable that the circle surface will appear dislocated. If this is the case; select it and press ALT−O to clear the origin (*Clearing origin*.).



Clearing origin.

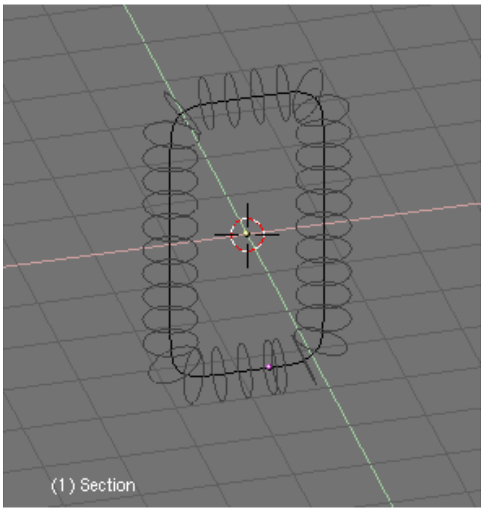
If you hit ALT−A the circle will follow the curve.

Now you probably will have to adjust the TrackX, Y, Z and UpX, Y, Z animation buttons, to make the circle go perpendicular to the curve path (*Tracking the right axis.*).



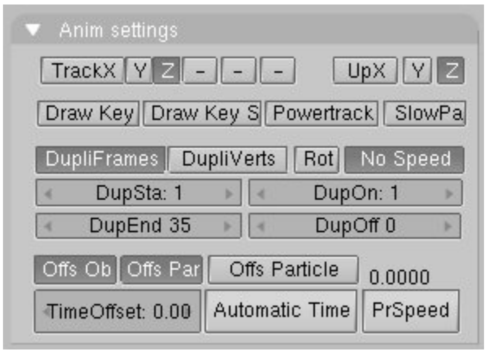
Tracking the right axis.

Now select the Surface Circle and go to Anim Settings Panel and press DupliFrames. A number of instances of the circular cross section will appear along the curve path (*DupliFrames!*).



DupliFrames!

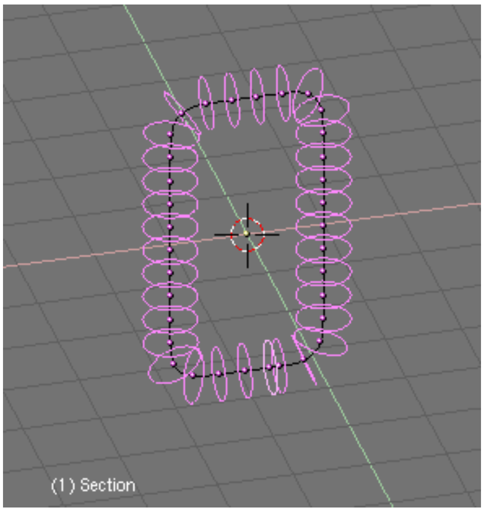
You can adjust the number of circles you want to have with the DupSta, DupEnd, DupOn and DupOff buttons. These buttons control the Start and End of the duplication, the number of duplicates each time and also the Offset between duplications. If you want the link to be opened, you can try a different setting for DupEnd (*Values for DupliFrames. Note "DupEnd: 35" will end link before curve's end.*).



Values for DupliFrames. Note "DupEnd: 35" will end link before curve's end.

To turn the structure into a real NURBS−object, select the Surface Circle and press CTRL−SHIFT−A.

A pop−up menu will appear prompting OK? Make Dupli's Real (*Making Dupli's Real.*).



Making Dupli's Real.

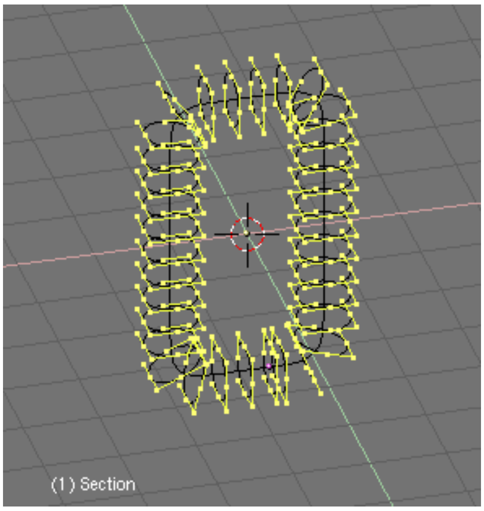
Do not deselect anything. We now have a collection of NURBS forming the outline of our object, but so far they are not skinned, so we cannot see them in a shaded preview or in a rendering.

To achieve this, we need to join all the rings to one object. Without deselecting any rings, press CTRL−J and confirm the pop−up menu request.

Now, enter Edit Mode for the newly created object and press A to select all vertices (*Skinning the link.*).

Now we are ready to skin our object.

Press F and Blender will automatically generate the solid object. This operation is called *Skinning* and is fully described in *Surface Curves Skinning*.



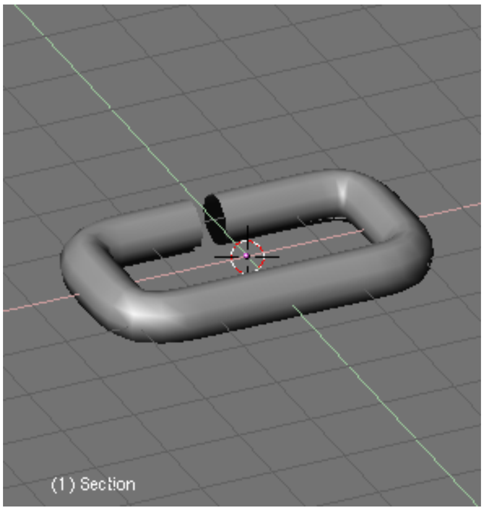
Skinning the link.

When you leave Edit Mode, you can now see the object in a shaded view. In some versons of Blender it may appear very dark. To correct this, enter Edit Mode and select all vertices, then press W. Choose Switch Direction from the menu and leave Edit Mode. The object will now be drawn correctly (*Skinned link.*).

The object we have created is a NURBS object. This means that you can still edit it. Even more interestingly, you can also control the resolution of the NURBS object via the Edit Buttons.

Here you can set the resolution of the object using ResolU and ResolV, so you can adjust it for working with the object in a low resolution, and then set it to a high resolution for your final render. NURBS objects are also very small in file size for saved scenes. Compare the size of a NURBS scene with the same scene in which all NURBS are converted (ALT−C) to meshes.

Finally you can delete the curve we used to give the shape of the link, since we no longer need it.

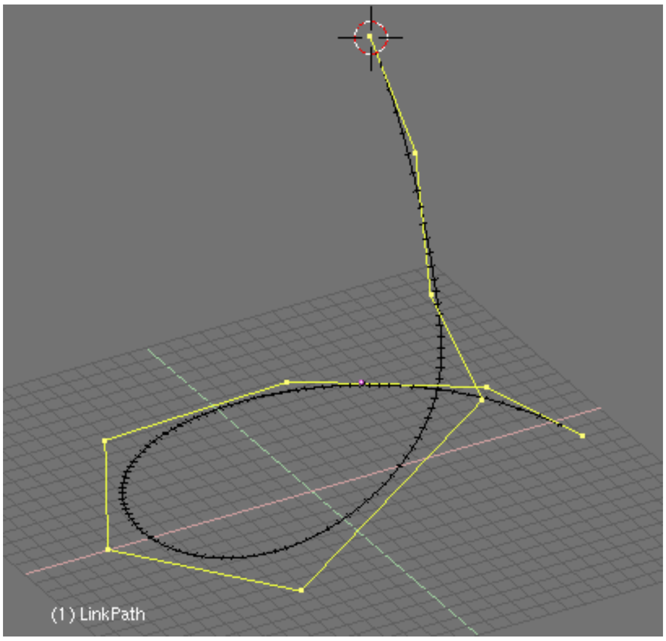


Skinned link.

Arranging objects with DupliFrames

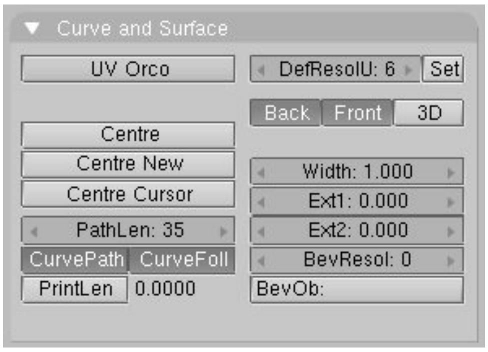
Now we will continue modelling the chain itself. For this, just add a Curve Path (we could use a different curve but this one gives better results). In Edit Mode, move its vertices until get the desired shape of the chain (*Using a curve path to model the chain.*). If not using a Curve Path, you should check the button 3D in the

Edit Buttons to let the chain be real 3D.



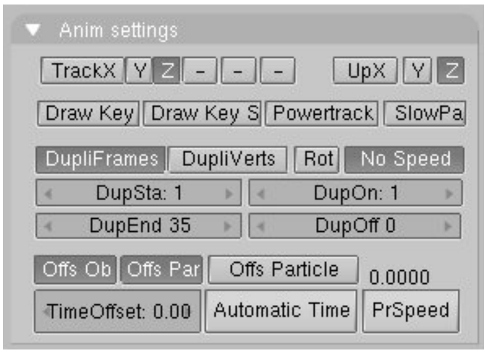
Using a curve path to model the chain.

Select the object "Link" we modelled in the previous step and parent it to the chain curve, again as a normal parent. Since we are using a Curve Path the option CurvePath in the AnimButtons will be automatically activated, however the CurveFollow option will not, so you will have to activate it (*Curve settings.*).



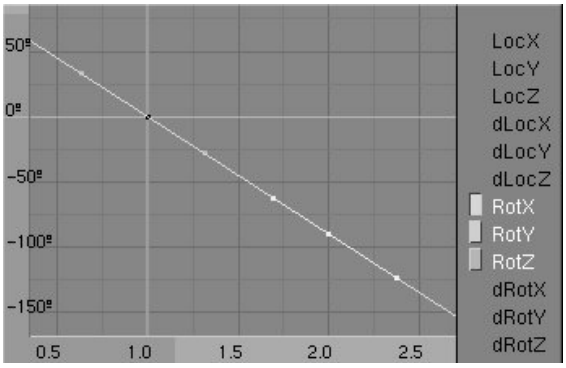
Curve settings.

If the link is dislocated, select it and press ALT−O to clear the origin. Until now we have done little more than animate the link along the curve. This can be verified by playing the animation with ALT−A. Now, with the link selected once again go to the Object Context and Anim settings Panel. Here, activate the option DupliFrames as before. Play with the DupSta:, DupEnd: and DupOf: NumButtons. Normally we are going to use DupOf: 0 but for a chain, if using DupOf: 0 the links are too close from each other you should change the value PathLen for the path curve to a lesser value, in the Editing Context and Curve and Surface Panel and then correspondingly change the DupEnd: value for the link to that number (*Adjusting the DupliFrames.*).



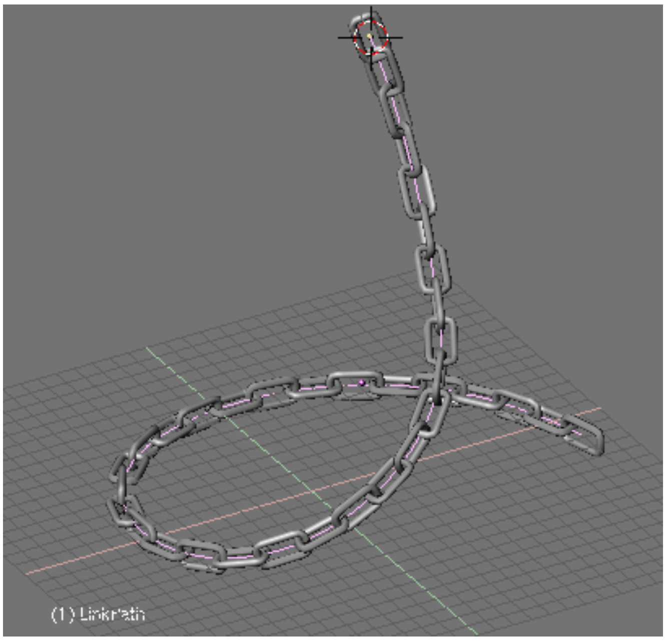
Adjusting the DupliFrames.

We need it so that the link rotates along the curve animation, so we have each link rotated 90 degrees with respect to the preceding one in the chain. For this, select the link and press Axis in the Edit Buttons to reveal the object's axis. Insert a rotation keyframe in the axis which was parallel to the curve. Move 3 or 4 frames ahead and rotate along that axis pressing R followed by X−X (X twice), Y−Y, or Z−Z to rotate it in the *local* X, Y or Z axis (*Rotating the link.*).



Rotating the link.

Open an IPO window to edit the rotation of the link along the path. Press the Extrapolation Mode so the link will continually rotate until the end of the path. You can edit the IPO rotation curve to make the link rotate exactly 90 degrees every one, two or three links (each link is a frame). Use N to locate a node exactly at X=2.0 and Y=9.0, which correspond to 90 degrees in 1 frame (from frame 1 to 2). Now we got a nice chain (*Dupliframed chain.*)!



Dupliframed chain.

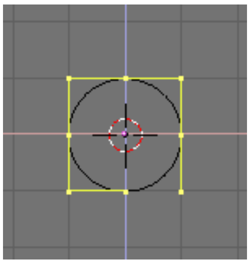
More Animation and Modelling

You are not limited to use Curve Paths to model your stuff. These were used just for our own convenience, however in some cases there are no need of them.

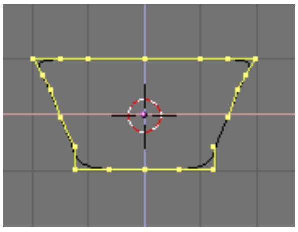
In Front View add a surface circle (you should know why by now *A Surface Circle.*).

Subdivide once, to make it look more like a square.

Move and scale some vertices a little to give it a trapezoid shape (*Trapezoidal cross−section.*).



A Surface Circle.

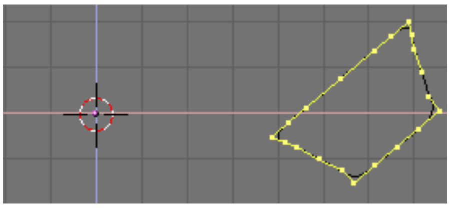


Trapezoidal cross−section.

Then rotate all vertices a few degrees.

Grab all vertices and displace them some units right or left in X (but at the same Z location). You can use CTRL to achieve this precisely.

Leave Edit Mode (*Trapezoidal cross section, rotated and translated.*).



Trapezoidal cross section, rotated and translated.

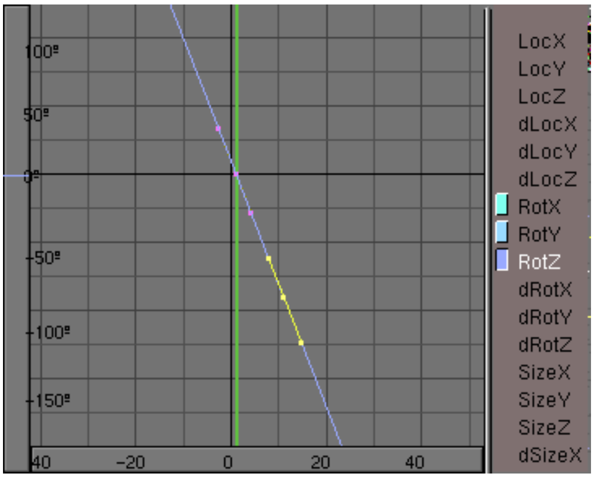
From now on, the only thing we are going to do is editing IPO animation curves. So you can call this "Modelling with Animation" if you like. We will not enter Edit Mode for the surface any more.

Switch to Top View.

Insert a KeyFrame for rotation at frame 1, go ahead 10 frames and rotate the surface 90 degrees over its new origin.

Insert one more KeyFrame.

Open an IPO window, and set the rotation IPO to Extrapolation Mode (*Rotation IPO for the cross section.*).



Rotation IPO for the cross section.

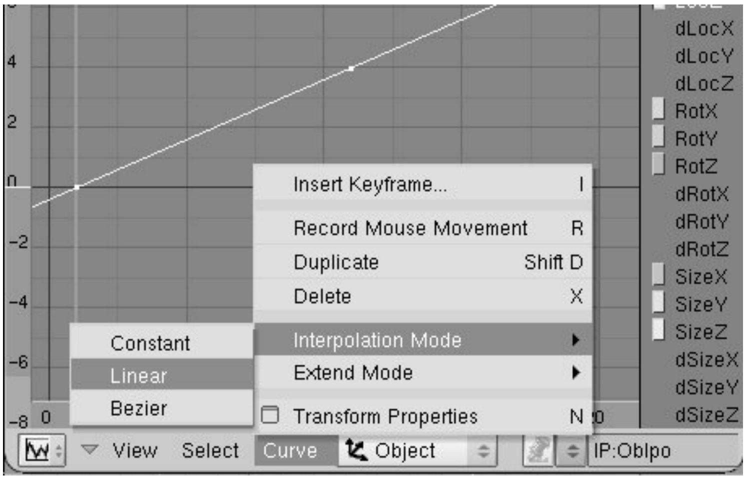
Go back to frame 1 and insert a keyframe for Location.

Switch to Front View.

Go to frame 11 (just press UPARROW) and move the surface in Z a few grid units.

Insert a new keyframe for Location.

In the IPO window set the LocZ to Extrapolation Mode (*Translation IPO for the cross section.*).



Translation IPO for the cross section.

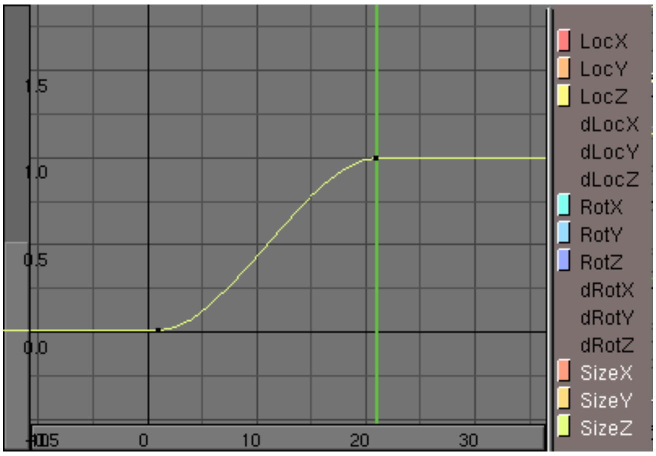
Now, of course, go to the Animation buttons and press DupliFrames. You can see how our surface is ascending in a spiral through the 3D space forming something like a spring. This is nice, however we want more. Deactivate DupliFrames to continue.

In frame 1 scale the surface to nearly zero and insert a keyframe for Size.

Go ahead to frame 41, and clear the size with ALT−S.

Insert a new keyframe for size.

This IPO will not be in extrapolation mode since we don't want it scaled up at infinitum (*Size IPO for the cross section.*)?



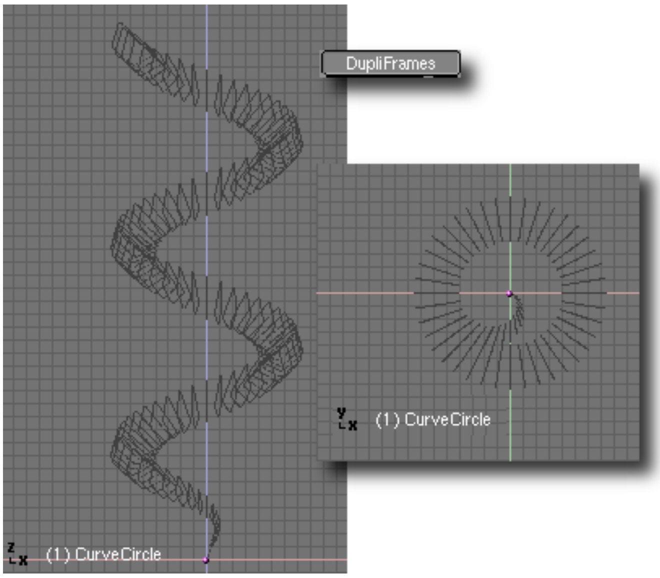
Size IPO for the cross section.

If you now activate DupliFrames you will see a beautiful outline of a corkscrew (*Using a curve path to model the chain.*).

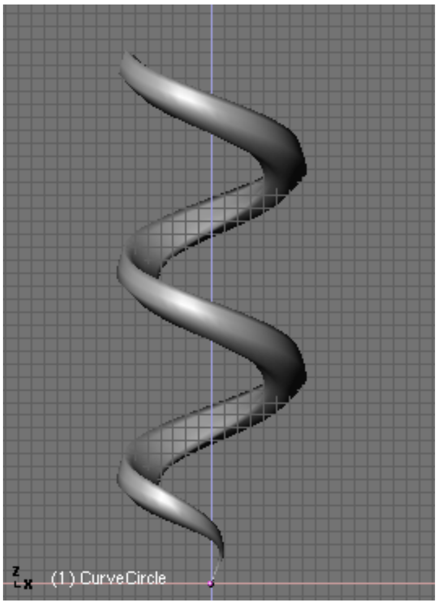
Once again the last steps are:

- Make Duplis Real
- Joining the surfaces
- Select all vertices and skinning
- Switch direction of normal if needed

- Leave Edit Mode (*Using a curve path to model the chain.*).



Using a curve path to model the chain.



Using a curve path to model the chain.

You can see this was a rather simple example. With more IPO curve editing you can achieve very interesting and complex models. Just use your imagination.

Previous: Manual/PartXIII/DupliVerts	Contents	*Next:* Manual/PartXIV

Use this navigator bar to follow the flow of the main page index for the Blender user manual:

Previous: Manual/DupliVerts	Contents	*Next:* Manual/Modelling Scripts

User Manual: Contents | Guidelines | Blender Version **2.4**

Main Section for Scripts

There are two main sets of documentation of the Blender scripts:

- Scripts/Manual: Contains Tutorials on using Python. Links to script information, Blender Python API docs.
- Scripts/Catalog: a large listing of scripts, grouped by function, linking to individual script pages.

Search the Catalog to find and download the script you need.

- Some Modeling Script Samples

Poly Reducer

Mode: Edit Mode (Mesh)

Menu: Mesh Scripts Poly Reducer

Description

This tool can be used instead of Blenders decimator modifier as a way to remove polygons from a mesh while keeping the original shape as much as possible.

Reasons you may want to use a polygon reducer are:

- To make 3D Scanned data usable when rendering and editing.
- Generate Level Of Detail models (LOD's), for games or simulation models.
- To speed up render times.

Options

Poly Reduce is accessed from Edit Mode and will operate on the entire mesh.

On activation a popup will be appear with the following options.

Poly Reduce
: Scale the meshes poly count by this value.

Boundary Weight
: Weight boundary verts by this scale. Zero disables boundary weighting. A boundary vert is a vert that is not completely surrounded by faces. Some meshes have no boundary verts. eg. a cube has no boundary verts where a plane has all boundary verts.

Area Weight
: Collapse edges affecting lower area faces first. Zero disables area weighting.

Triangulate

Convert quads to tris before reduction, for more choices of edges to collapse.

The advantage of triangulating is you have a larger set of edges to choose from when collapsing giving a higher quality result.

UV Coords
: Interpolate UV Coords (if existing)

Vert Colors
: Interpolate Vertex Colors (if existing)

Vert Weights
: Interpolate Vertex Weights. (if existing)

Hints

Poly reducer has some advantages and disadvantages compared to Blenders decimator modifier, here are some pros and cons.

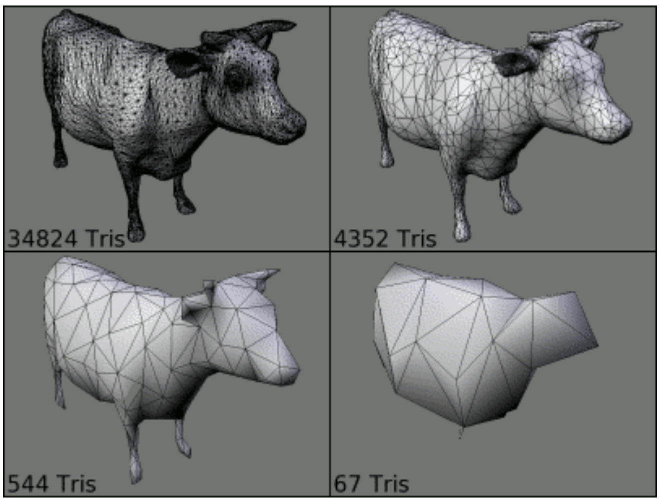
Pros

- Higher quality resulting mesh.
- Can operate on any mesh, will not throw errors if the mesh has odd face/edge/vert topology.
- Options to control where polygons are removed.
- Keeps materials assigned to faces.
- Maintains UV Texture coordinates, Vertex colors, and Vertex Group Weights (used for bone weight painting) − This makes it very useful for game/realtime models.

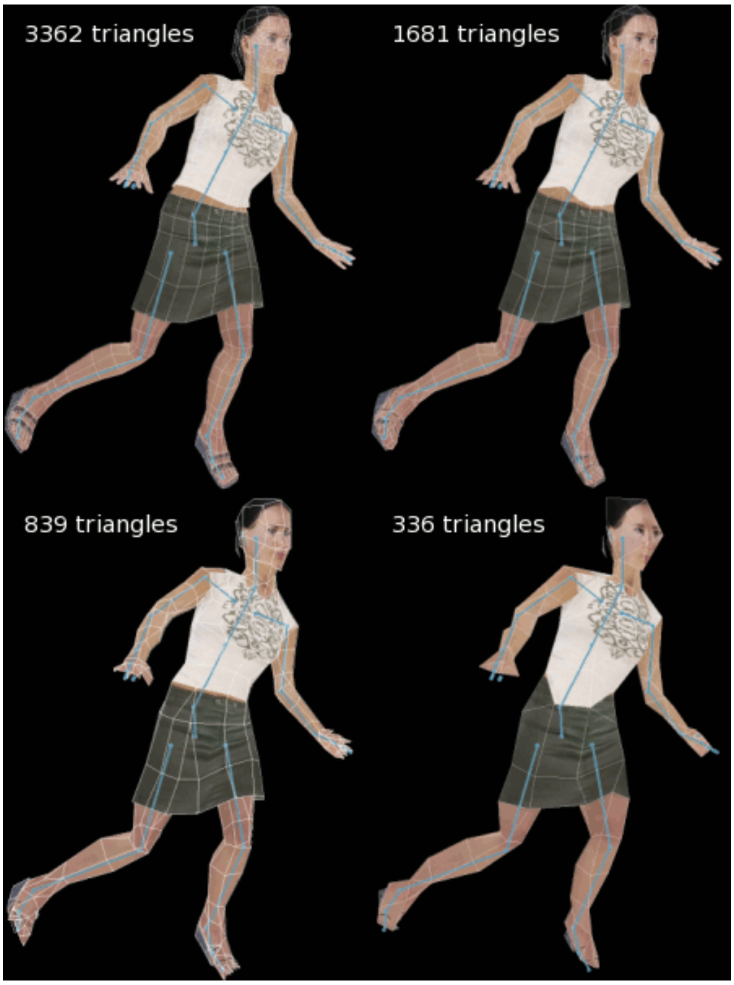
Cons

- Fairly Slow
- Uses a lot of memory

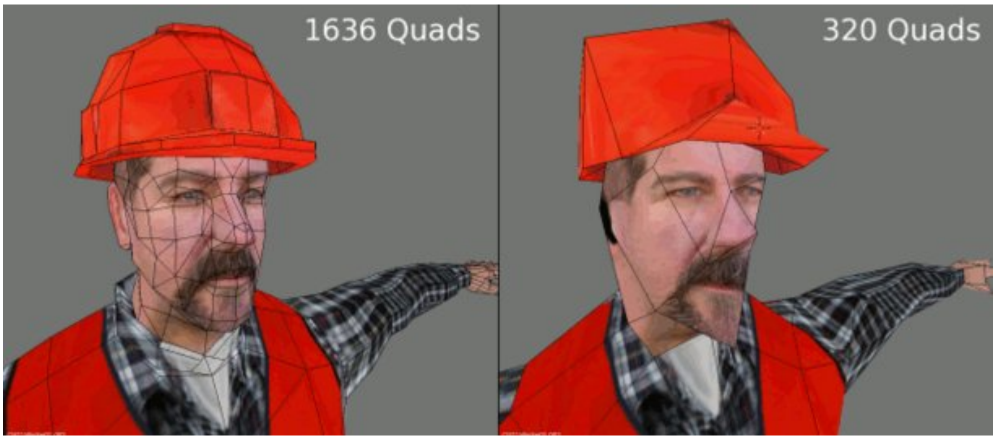
Examples



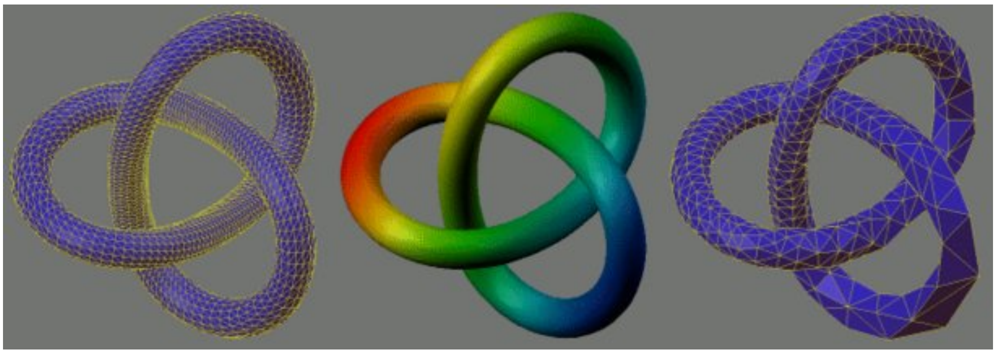
famous cow.



Human with UV textures and bone weights from [http://www.x−trusion.com](http://www.x-trusion.com)



Heavily reduced workman [http://www.x−trusion.com](http://www.x-trusion.com)



Example of an 80% Reduction using a weight map for influencing the result− Original, Weight Map, Result of or an

Auto Image Layout

Mode: All Modes (Mesh)

Menu: UV/Image Editor UVs Auto Image Layout

Description

This script makes a new image from the used areas of all the images mapped to the selected mesh objects.

Image are packed into 1 new image that is assigned to the original faces.

This is useful for game models where 1 image is faster than many, and saves the labour of manual texture layout in an image editor.

Options

This script is accessed from UV/Face mode and packs images from the active mesh.

On activation a popup will be appear with the following options.

image path
: no ext
: A new PNG image file will be created at this path. use // as a prefix for the current blend file location. otherwise you may specify the full path. Do not add in a file extension.

Pixel Size
: The size of the image, this value is used for width and height to make a square image.

Pixel Margin
: When cropping the image to the bounds of the used areas add this pixel margin, this stops lower resolution textures (mipmaps) from bleeding the edge colour into the faces that use this texture.

Keep Image Aspect

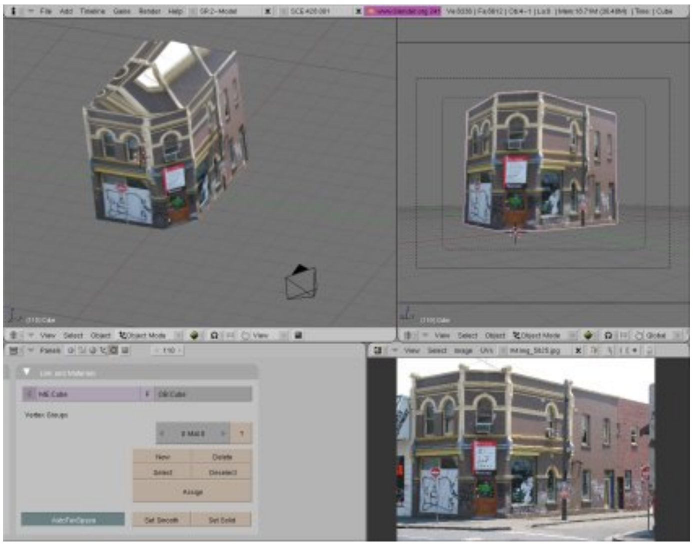
If this is turned off, the tiles will stretch to the bounds of the image, making the images look stretched in an image viewer. however it will give better results when viewed in 3d because there is more pixel information in the image.

Texture Source, All Sel Objects

When enabled all selected objects will have their textures packed into the texture.

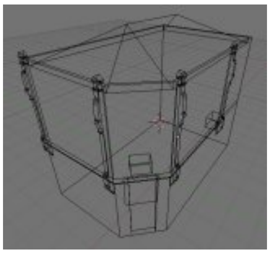
Examples

Here is an test case where I took 5 unedited photos, mapped them to a low poly mesh, and pack them into 1 texture.

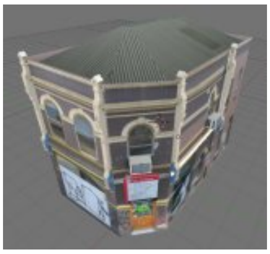


Projection mapped uv mesh

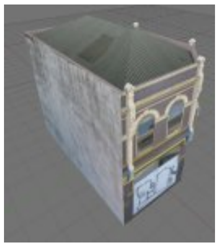
Model in details



Finished Details with roof and side walls



Texture view



Back wall with generic texture

All images used for this mesh

Result of running the Auto Texture Layout Script

Weight Painting

Bone Weight Copy

Mode: Object Mode (Mesh)

Menu: Object Scripts Bone Weight Copy

Description

This copies weights from one mesh to another based on vertex locations. It can also be used to update a mesh that's already weighted, by selecting the verts on the target mesh. Then using the "Copy To Selected" option.

Mirror Vertex Locations & Weight

Mode: Edit Mode (Mesh)

Menu: Mesh Scripts Mirror Vertex Locations & Weight

Description

This script is used to mirror vertex locations and weights. It is useful if you have a model that was made symmetrical but has verts that have moved from their mirrored locations slightly, causing Blender's X−Mirror options not to work.

Weights can be mirrored too, this is useful if you want to model 1 side of a mesh, copy the mesh and flip it. You can then use this script to mirror to the copy, even creating new flipped vertex groups, renaming group name left to right or .L to .R

Vertex positions are mirrored by doing a locational lookup, finding matching verts on both sides of a mesh and moving to the left/right or mid location.

The vertex weights work differently, they are mirrored by location also, but they mirror in pairs, rather it works by finding the closest vertex on the flip side and using its weight.

When a location mirror is finished, verts that have not been mirrored will remain selected. A good way to check both sides are mirrored is to select the mirrored parts, run this script with default options and then see of there are any selected verts.

For details on each option read the tooltips.

Weight Paint Gradient

Mode: Weight Paint (Mesh)

Menu: Paint Weight Gradient

Description

Mix weight paint and face select mode so as to select the faces to gradient. Then Run "Gradient" from the weight paint menu, and click on the 2 locations to blend between. The existing weight under the mouse is used for to/from weights.

Vertex Colour Gradient

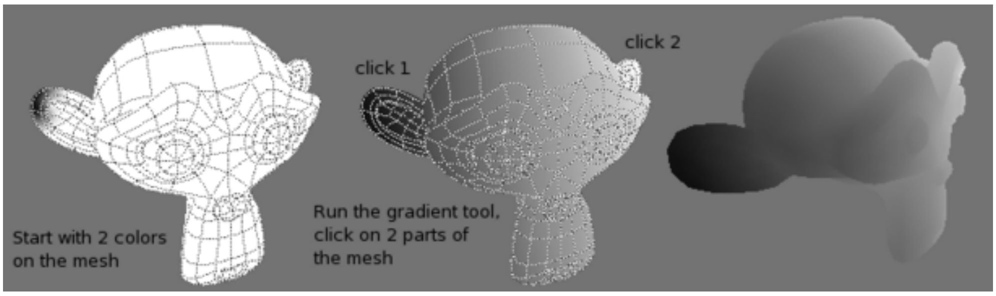
Mode: Vertex Paint (Mesh)

Menu: Paint VCol Gradient

Description

see Weight Paint Gradient

Examples



Example of gradient usage

Self Shadow

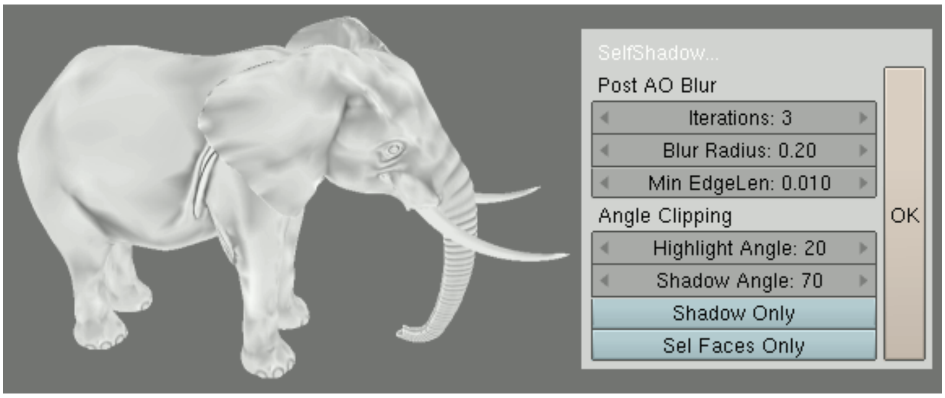
Mode: Vertex Paint (Mesh)

Menu: Paint Self Shadow VCols (AO)

Description

Uses the mesh geometry to shade the mesh, similar to Ambient Occlusion.

Examples



Elephant Shaded

Previous: Manual/DupliFrames | Contents | *Next:* Manual/Modifiers & Deformation

User Manual: Contents | Guidelines | Blender Version **2.43**

Modifiers are automatic operations that affect an object in a non−destructive way. With modifiers, you can perform many effects automatically that would be otherwise tedious to do manually (such as subdivision surfaces) and without affecting the base topology of your object. Modifiers work by changing how an object is displayed and rendered, but not the actual object geometry. You can Apply a modifier if you wish to make it's changes permanent.

There are two types of modifiers:

- Deform modifiers

  Deform modifiers only change the shape of an object, and are available for Mesh, Text, Curve, Surface and Lattice objects.

- Constructive modifiers

  Constructive modifiers are only available for Mesh objects, and typically change mesh topology in some way (e.g. the subsurf modifier which performs Catmull−Clarke subdivision).

Interface

The interface that is used for modifiers and constraints is described here:

- Modifier Stack

Modifiers

Mode: Any Mode

Panel: Editing Context Modifiers

Hotkey: F9 (Panel)

Modifiers are added from the Modifiers tab in the Edit Buttons (F9). The Modifiers tab appears when a Mesh, Curve, Surface, Text, or Lattice Object is added or selected.

- **Armature** − Use bones to deform and animate your mesh.
- **Array** − Create an array out of your basic mesh and similar (repeating) shapes.
- **Bevel** − Create a bevel on a selected mesh/object.
- **Booleans** − Combine/subtract/intersect your mesh with another one.
- **Build** − Assemble your mesh step by step when animating.
- **Cast** − Shift the shape of a mesh, surface or lattice to an sphere, cylinder or cuboid.
- **Cloth** − Simulate the properties of a piece of cloth. It is inserted in the modifier stack when you designate a mesh as Cloth.
- **Curve** − Bend your mesh using a curve as guide.
- **Decimate** − Reduce the polygon count of your mesh.
- **Displace** − Use textures or objects to displace your mesh.
- **EdgeSplit** − Add sharp edges to your mesh.

- **Explode** − Splits apart a mesh when, used with particles.
- **Hooks** − Add a hook to your vertice(s) to manipulate them from the outside.
- **Lattice** − Use a Lattice object to deform your mesh.
- **Mirror** − Mirror an object about one of its own axis, so that the resultant mesh is symmetrical, and you only have to model/edit half or a fourth of it.
- **Particle Instance** − Make an object act similar to a particle but using the mesh shape instead.
- **Smooth** − Smooth a mesh by flattening the angles between its faces.
- **SubSurf** − Smooth the surface by creating interpolated geometry.
- **UVProject** − Project UV coordinates on your mesh.
- **Wave** − Deform your (dense) mesh to form an (animated) wave.

Previous: Manual/Modelling Scripts | Contents | *Next:* Manual/Modifier Stack

User Manual: Contents | Guidelines | Blender Version **2.4x**

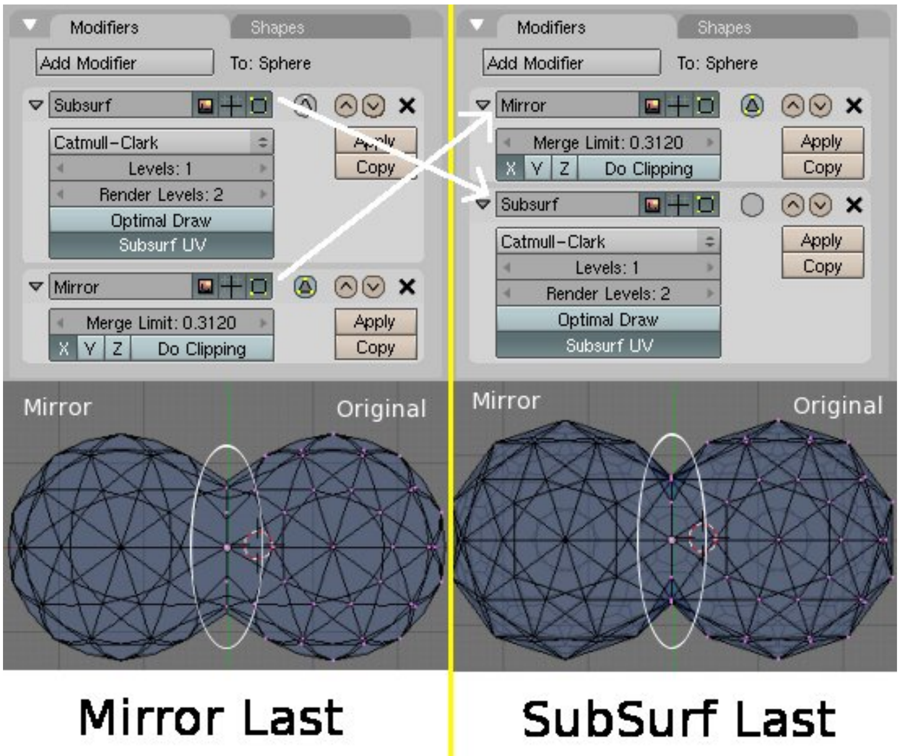
Modifier

A Modifier is defined as the application of a "process or algorithm" upon Objects. They can be applied interactively and non−destructively in just about any order the users chooses. This kind of functionality is often referred to as a "modifier stack" and is found in several other 3D applications.

Modifiers are added from the Modifiers tab in the edit buttons (F9). The Modifiers tab appears when a Mesh, Curve, Surface, Text, or Lattice Object is added or selected.

Some "tools", for example the "Decimator", have been migrated from its previous location and changed into a modifier.

In a modifier stack the order in which modifiers are applied has an effect on the result. Fortunately modifiers can be rearranged easily by clicking the convenient up and down arrow icons. For example, (*Stack ordering*) shows SubSurf and Mirror Modifiers that have switched places.



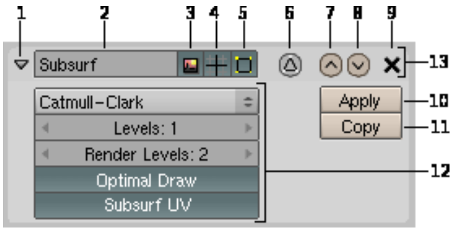
Stack ordering

The left side of the Yellow line has the Mirror Modifier as the last item on the stack. In the lower half you can see two spheres. One is a "mirror" of the other. The original sphere is on the right and the "mirror" on the left. The Mirror Modifier's Merge Limit: has been set to a value of "**0.312**" to cause the vertices to weld together from a greater distance. The area marked by a white circle is a suggested area to concentrate on when the stack order is changed later.

The right side of the Yellow line has the SubSurf Modifier "Switched/Rearranged" to the bottom of the stack (i.e. it has switched places with the Mirror Modifier). Now take a look at the white circle area. You can see that the results looks different than previously. This means that the stack order is very important in defining the end results. In this case the SubSurf Modifier is being applied last.

Interface

Each Modifier has been brought in from a different part of Blender, so each has its own unique settings and special considerations. However, each Modifier's interface has the same basic components, see (*Panel Layout (Subsurf as an example)*).



Panel Layout (Subsurf as an example)

- 1 ( ) − Collapses modifier to show only the header.
- 2 − A name for this modifier; the default is the name of the modifier itself. It is unique amongst other modifiers of the same type.
- 3 ( ) − Shows modifier effect in the rendering view.
- 4 ( ) − Shows modifier effect in the 3D view.
- 5 ( ) − Shows modifier effect in Edit mode. This button may not be available depending on the type of modifier.
- 6 (, ,) − Applies modifier to editing cage in Edit mode. The icon can be Disabled, Deactivated and Activated, respectively. This icon is Cage Mode.
- 7 ( ) − Moves modifier up in the stack.
- 8 ( ) − Moves modifier down in the stack.
- 9 ( ) − Removes the modifier from the stack.
- 10 (Apply) − Makes the modifier real.
- 11 (Copy) − Creates a copy of the modifier at the base of the stack.
- 12 (Sub−panel) − Sub panel for individual modifiers.
- 13 − Header area for the main modifier controls.

Every modifier features a Collapse Arrow (**1**) and Name Box (**2**). The Collapse Arrow hides the modifier's Sub−panel (**12**) so that multiple modifiers can be displayed without the need for excessive scrolling in the buttons window. When collapsed, only the modifier Header (**13**) is displayed. The Name Box can be used to give your modifiers titles, which can make recognizing their functions easier. This comes in handy in large scenes with complex modifier setups that feature multiple modifier types.

They are followed by three buttons which control the visibility of the effect in three separate contexts: Rendering (**3**), Object Mode (**4**), and Edit Mode (**5**). Toggling each button determines whether the modifier's result displays in each mode.

The Cage Mode (**6**) button is used to apply the modifier to the editing cage, which generates a more accurate display of the underlying geometry once a modifier has been applied. This displays vert/edge/face positions in their "modified" locations instead of their original locations. It should be noted, however, that transformation operations still act on the original locations of the cage vertices and not on the displayed locations. This button has three states: Disable, Activated and Deactivated. If it is Disabled then the modifier is not permitting the edit cage to changed.

The two Arrow buttons (**7** and **8**) control the order in which modifiers exist in the stack. Modifiers are evaluated top to bottom in the panel. The higher in the panel, the earlier it is evaluated. This can be very important depending on the application.

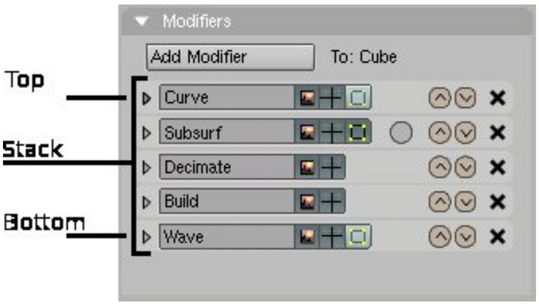
A great example of a situation in which positioning in the stack is important is the use of the subsurf modifier in combination with the mirror modifier as shown in (*Stack Ordering*) from the previous section. The mirror modifier must appear before the SubSurf modifier in the list, otherwise the original half of the object gets subsurfaced and **then** mirrored, which may not be what you expected.

The Delete (**9**) button does exactly what one would expect; it removes the modifier from the stack entirely.

The Apply (**10**) and Copy (**11**) buttons have two very different functions despite their proximity to each other. Apply evaluates the modifier as if it were the first modifier in the stack and writes the results into the mesh, in effect "baking" the result of that modifier into the object. The Copy button creates a copy of the current modifier, including its settings, at the bottom of the modifier list.

Stack

To add a Modifier you add it to the Stack. Once added they can be rearranged under most conditions.



Stack Example

Some Modifiers can't be rearranged in the stack because they rely on certain information from the underlying Object data structure. These types of Modifiers are static in the stack and always insert themselves at the *Top* relative to the panel's point of view. See (*Stack Example*).

For example, The Lattice Modifier can not be moved from the *Top* because it requires the original Object data. When you attempt to move it *Down* in the stack you will get the error message: "**Cannot move beyond a non−deforming modifier**". And like wise, if you attempt to move a modifier *Above* the Lattice Modifier you will get the error message: "**Cannot move above a modifier requiring original data**".

Hence, if a modifier places itself at the *Top* of the stack it means the modifier requires the Original Object data, which is only available at the *Top*.

Some Modifiers can only be applied to certain Object types. This is indicated by the panel filtering the "Add modifier" button on the Modifiers panel. Only modifiers that can be applied are shown in the listbox button.

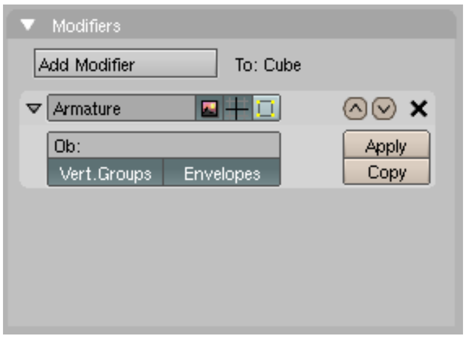
For example, Mesh objects can have all available Modifiers applied. But Lattice type objects can only have: Lattice, Curve, Hooks, Wave and Armature Modifiers applied.

Previous: Manual/Modifiers & Deformation	Contents	*Next:* Manual/Armature Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

This sub−panel appears in the Editing Context panel group which is accessed using F9 or clicking button in the Buttons window. This sub−panel is part of the Modifier parent panel. For further information about the common panel components see the Interface section on modifiers.

Armature Modifier



Modifier panel with Armature modifier activated.

- OB: − The name of the object to which this modifier will be applied.
- Vert.Groups − Enable/Disable vertex groups defining the deformation.
- Envelopes − Enable/Disable bone envelopes defining the deformation.

The Armature Modifier is used for building skeletal systems for animating the poses of characters and anything else which needs to be posed.

By adding an armature system to an object, that object can be deformed accurately so that geometry doesn't have to be animated by hand. The Armature Modifier allows objects to be deformed by bones simply by specifying the name of the armature object.

Previous: Manual/Modifier Stack | Contents | *Next:* Manual/Array Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

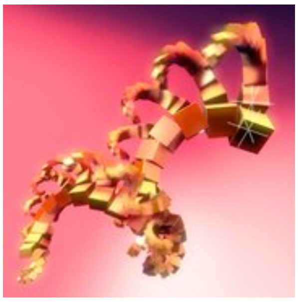
Array Modifier

Mode: Any Mode

Panel: Editing Context Modifiers

Hotkey: F9

Description

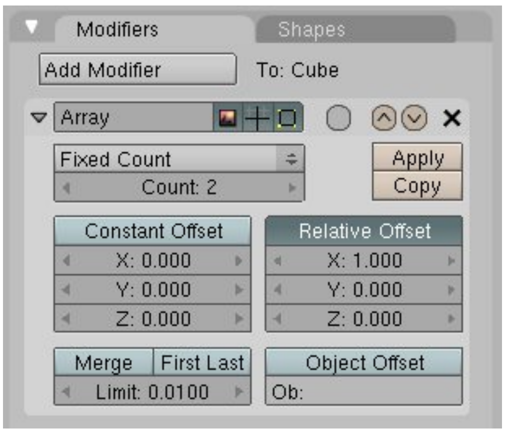


Multidimensional array animated with motion blur

The Array modifier creates an array of copies of the base object, with each copy being offset from the previous one in a number of possible ways. Vertices in adjacent copies can be merged based on a merge distance, allowing smooth subsurf frameworks to be generated.

This modifier can be useful when combined with tilable meshes for quickly developing large scenes. It is also useful for creating complex repetitive shapes.

Options



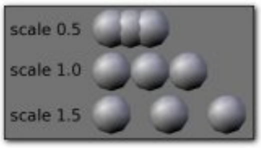
Array modifier

Constant Offset, X, Y, Z

Adds a constant translation component to the duplicate object's offset. X, Y and Z constant components can be specified.

Relative Offset, X, Y, Z

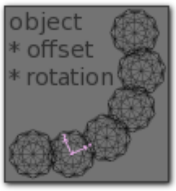
Adds a translation equal to the object's bounding box size along each axis, multiplied by a scaling factor, to the offset. X, Y and Z scaling factors can be specified. See (*Relative offset example*).



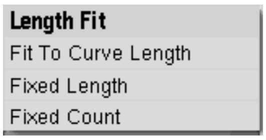
Relative offset example

Object Offset, Ob

Adds a transformation taken from an object (relative to the current object) to the offset. See (*Object offset example*).



Object offset example



Length Fit menu

Length Fit menu

Controls how the length of the array is determined; see (*Length Fit menu*). There are three choices. One of Ob, Length or Count will be displayed to allow entry of the appropriate parameter depending on the choice selected:

◊ Fit To Curve Length − Generates enough copies to fit within the length of the curve object specified in Ob

Note

Fit To Curve Length uses the local coordinate system length of the curve, which means that scaling the curve in Object mode will not change the number of copies generated by the Array modifier. Applying the scale (Ctrl A) can be useful in this case.

◊ Fixed Length − Generates enough copies to fit within the fixed length given by Length
◊ Fixed Count − Generates the number of copies specified in Count

Note

Both *Fit To Curve Length* and *Fixed Length* use the local coordinate system size of the base object, which means that scaling the base object in Object mode will not change the number of copies generated by the Array modifier. Applying the scale (Ctrl A) can be useful in this case.

Ob

The Curve object to use for Fit To Curve Length.

Length

The length to use for Fixed Length.

Count

The number of duplicates to use for Fixed Count.

Merge

If enabled, vertices in each copy will be merged with vertices in the next copy that are within the given merge distance.

First Last

If enabled **and** Merge is enabled, vertices in the first copy will be merged with vertices in the last copy (this is useful for circular objects; see (*First Last merge example*)).

 Subsurf discontinuity caused by not merging vertices between first and last copies (First Last off)	 Subsurf discontinuity eliminated by merging vertices between first and last copies (First Last on)
First Last merge example	

Merge Dist

Controls the merge distance for Merge Verts.

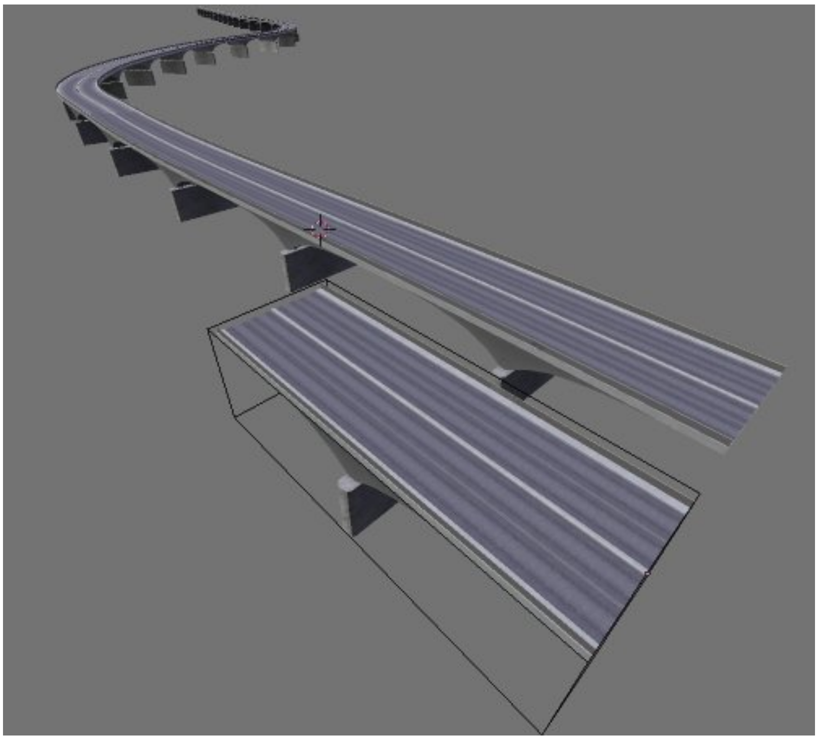
Hints

Offset Calculation

The transformation applied from one copy to the next is calculated as the sum of the three different components (Relative, Constant and Object), all of which can be enabled/disabled independently of the others. This allows, for example, a relative offset of (1, 0, 0) and a constant offset of (0.1, 0, 0), giving an array of objects neatly spaced along the X axis with a constant 0.1 unit of space between them.

Examples

Bridge

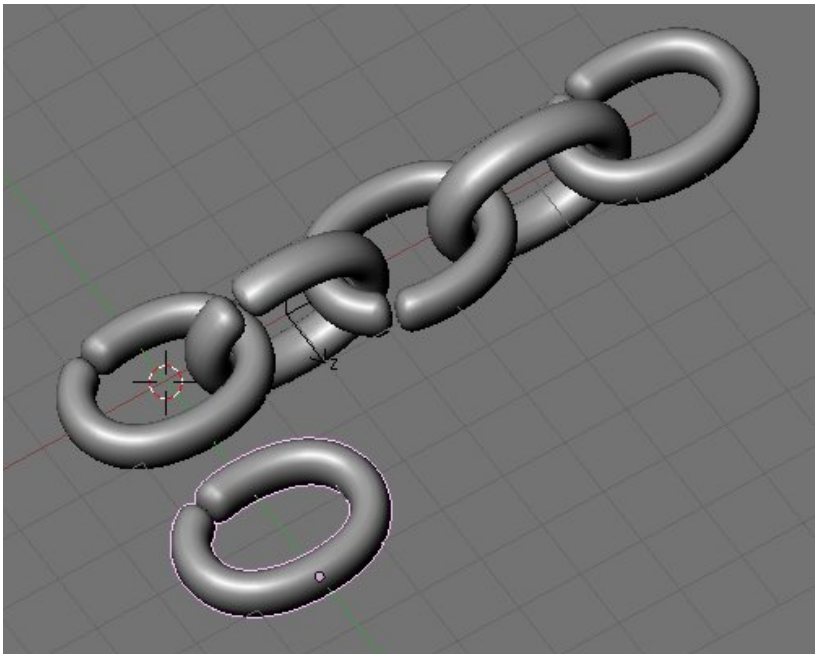


A bridge made from a tilable mesh

Note

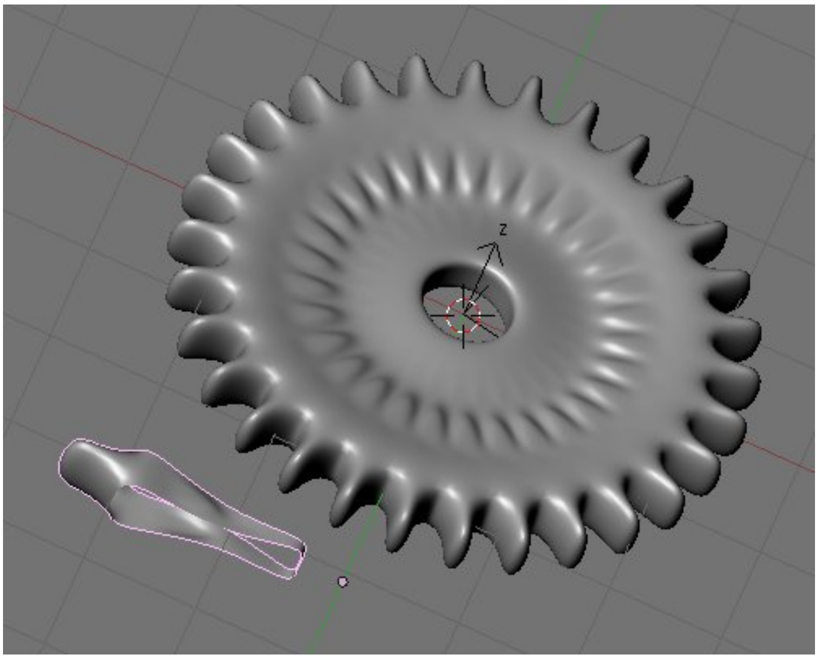
As the Curve modifier could not be after Array in the modifier stack at the time this image was created, the Array modifier was applied (i.e. the "Apply" button was pressed) before the curve was added in the bridge image.

Chain



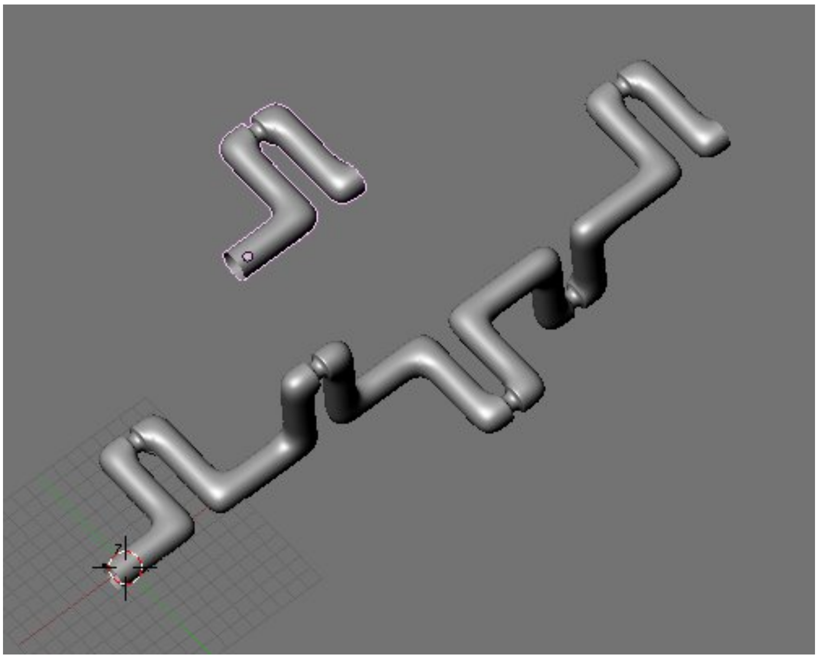
A chain created from a single link.
Sample blend file

Cog



A cog created from a single segment.
Sample blend file

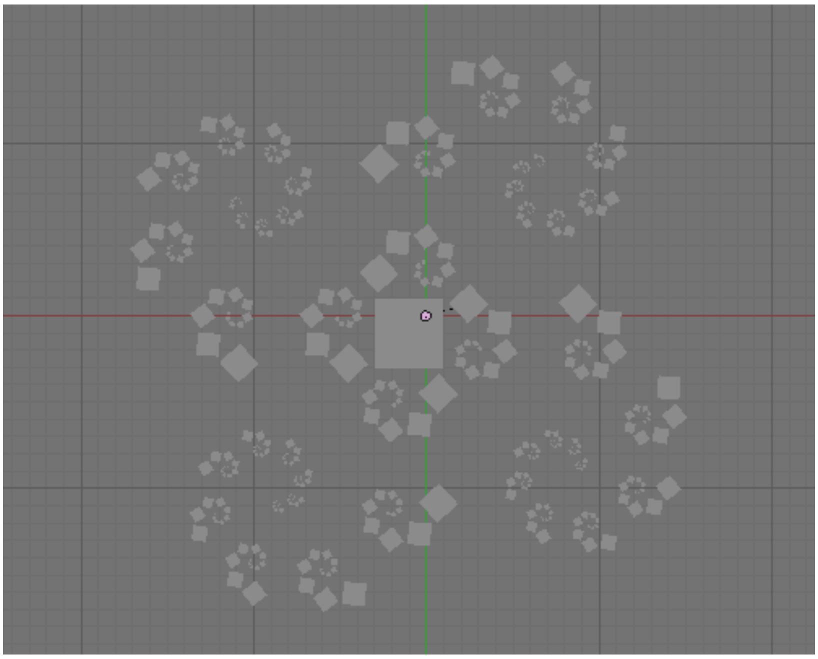
Crankshaft



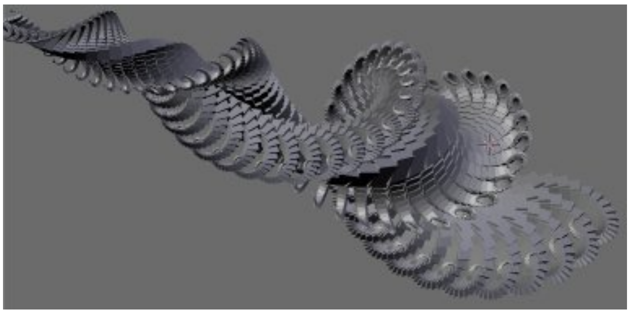
A crankshaft.
Sample blend file

Fractal

Multidimensional array animated with motion blur



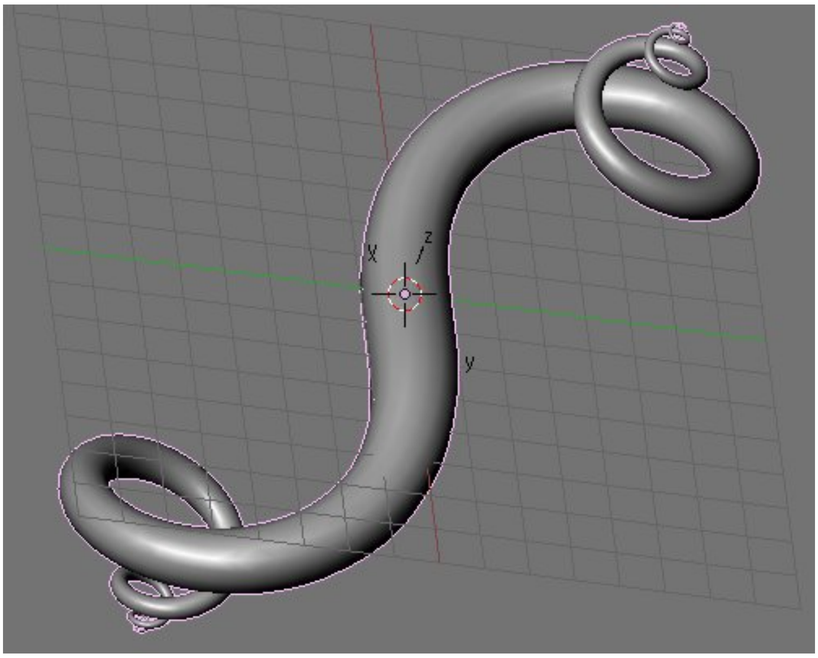
A fractal−like image created with multiple array modifiers applied to a cube.
Sample blend file



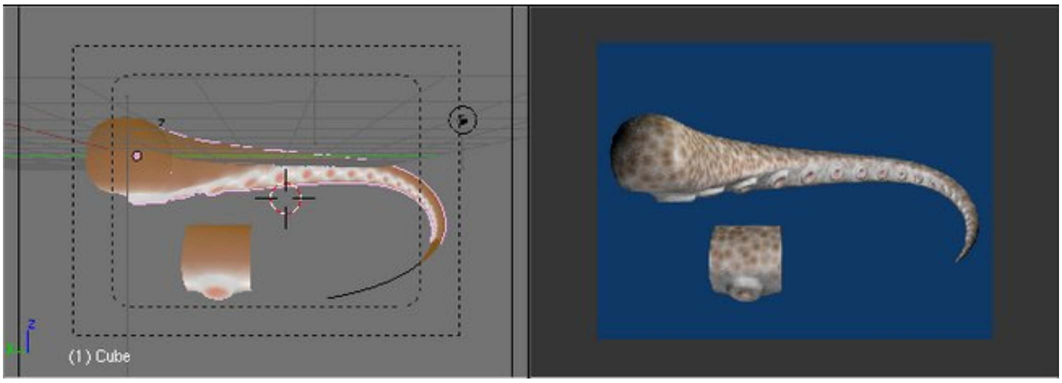
A fractal fern image created with 2 array modifiers and 1 mirror applied to a cube

Organic

Subsurfed cube array with 1 object offset, 4 cubes and a high vertex merge setting to give the effect of skinning

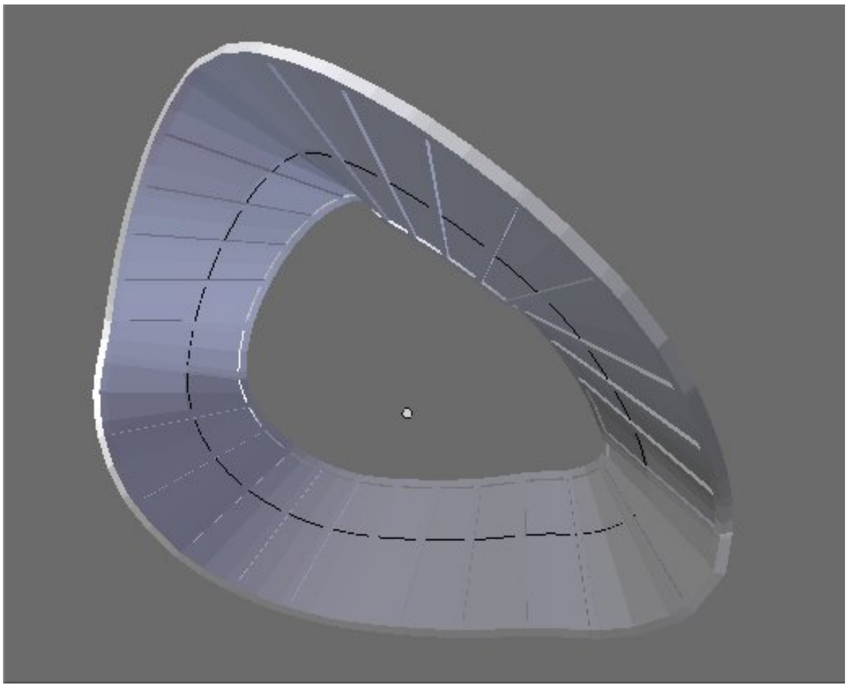


A double spiral created with two array modifiers and one subsurf modifier applied to a cube. As above, the vertex merge threshold is set very high to give the effect of skinning.
[Sample blend file]()



A tentacle created with an Array modifier followed by a Curve modifier. The segment in the foreground is the base mesh for the tentacle; the tentacle is capped by two specially modelled objects deformed by the same Curve object as the main part of the tentacle.
Sample blend file

Track



A track.
Sample blend file

Tutorials

Some tutorials that exploit the Array modifier:

- Creating A Double Helix With Modifiers

Previous: Manual/Armature Modifier	Contents	*Next:* Manual/Bevel Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

Bevel Modifier

Mode: Any Mode

Panel: Editing Context Modifiers

Hotkey: F9

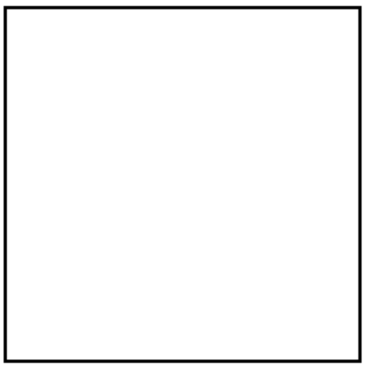
Description

The Bevel Modifier adds the ability to bevel the edges of mesh that the Bevel Modifier is applied to, allowing control of how and where the bevel is applied to the mesh.

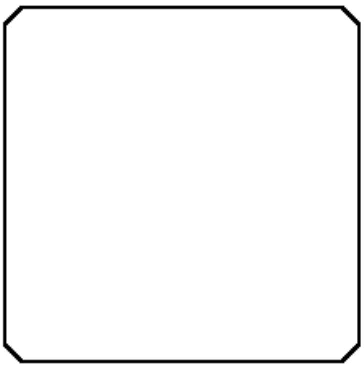
What is a Bevel?

The definition of a Bevel according to wikipedia.org is:

- A bevelled edge refers to an edge of a structure that is not perpendicular (but instead often at 45 degrees) to the faces of the piece. The words bevel and chamfer overlap in usage; in general usage they are often interchanged, while in technical usage they may sometimes be differentiated....



Unbevleled square



Bevelled square

The picture titled "Unbevelled square" shows a square which has unbevelled edges as the angles between the corners of the square are 90Â°/perpendicular. The picture titled "Bevelled square" shows a square which has bevelled corners.

Although the 2 pictures above show 2D squares the Blender Bevel Modifier can work on both 2D and 3D meshes of almost any shape, not just Squares and Cubes.

Default Bevel

The picture titled "Default Bevel" shows a Blender 3D cube which a bevel applied using just the default Bevel Modifier settings.

Setting Bevel Weight

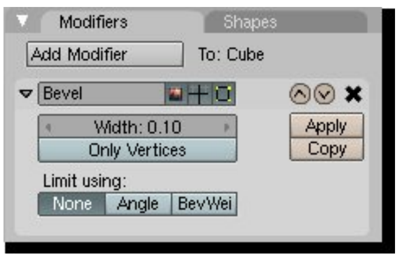
When certain bevel options (such as BevWeight in the Bevel Modifier) are set the mesh will need to have Bevel Weights assigned to it; otherwise the bevel will not be applied.

Bevel Weight can be applied to selected edges of a mesh by:

- Switching to Edit Mode.
- Selecting the edge or edges of the mesh that you wish to apply a Bevel Weight to.
- Pressing Shift Ctrl E or going to menu entry Mesh â†’ Edges â†’ Adjust Bevel Weight.
- Then by moving the mouse around or entering a value directly at the keyboard between −1 to +1 you should be able to change the default Bevel Weight. You will be able to see the current value of the Bevel Weight change at the Bottom status area of the 3D Viewport.
- If the Particular mesh already has a Bevel Modifier with the BevWeight button applied to it; as you alter the Bevel Weight you should be able to see that the bevel amount changes as the Bevel Weight is altered.

Options

The Bevel Modifier Panel is reasonably uncluttered panel and for the most part is intuitive. That said here is a description of some of the Buttons and Numeric Sliders contained within the panel:

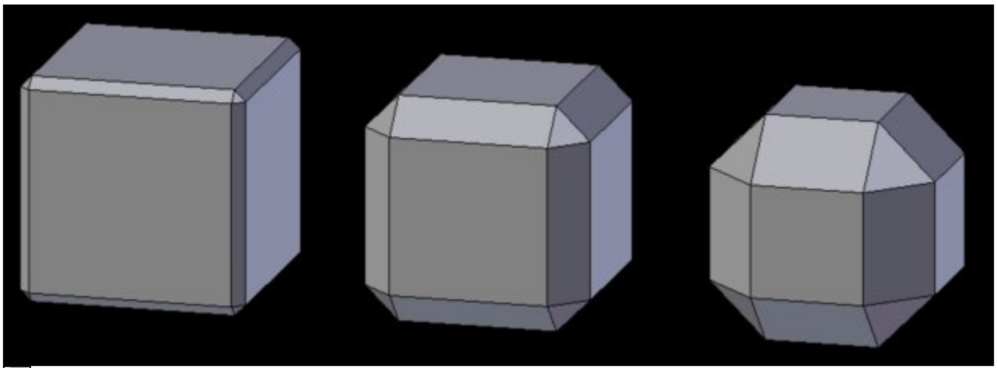


Bevel Modifier Panel

- Width

  The Width numeric slider entry field controls the width of the bevel applied to the base mesh. The Width can range from 0 (no bevel applied) to .5 (half a Blender Unit).

The picture titled "Bevel Width" shows 3 cube meshes each with a Bevel Modifier applied but having different Width values on each cube of .10, .30 and .50.

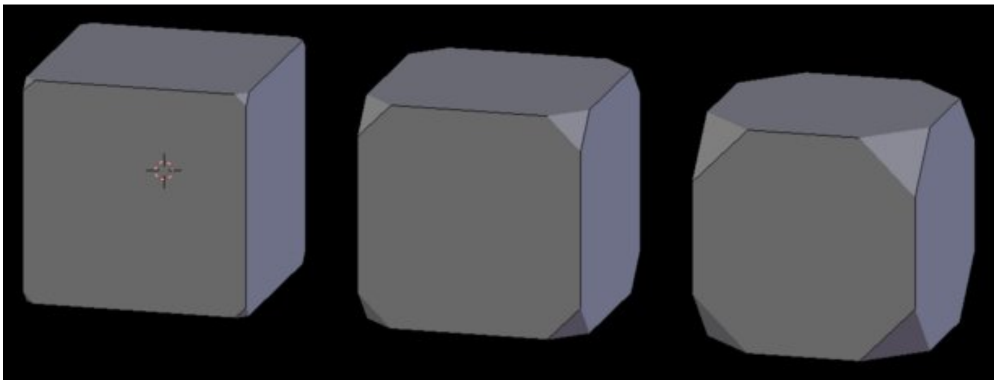


3 Cubes with .10, .30 & .50 Bevel Widths

- Vertices Only

The Vertices Only button in the Bevel Modifier panel alters the way in which a bevel is applied to the mesh. When the Vertices Only button is active only the areas near vertices are bevelled the rest of the face/edge is left unbevelled.

Below is a picture of 3 bevelled cubes but this time the Vertices Only button has been activated:



3 Cubes with .10, .30 & .50 Bevel Widths with Vertices Only option Button selected

- Limit Using:

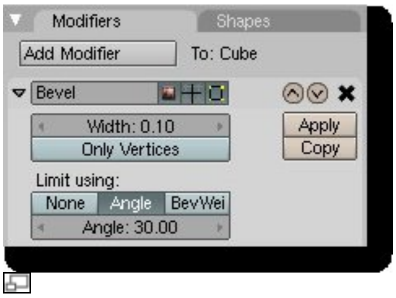
This section of the Bevel Modifier is used to control where and when a Bevel is applied to the underlying mesh.

- Limit Using: None

The None button in the Limit Using section controls if the Bevel Modifier applies a bevel to an entire mesh or not. If the None button is selected then no limits are put on where a bevel is applied and the entire underlying mesh will be bevelled.

- Limit Using: Angle

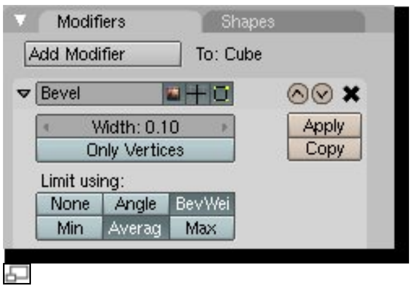
The Angle button in the Limit Using section when selected displays a numeric slider called Angle. This Angle slider is used to set the angle above which an edge will be bevelled. When the angle between meeting edges is less than the angle in the slider box the bevel on those specific edges will not be applied; however a bevel will be applied on other edges which are greater than the specified Angle.



Bevel Modifier with Angle limit displayed

- Limit Using: BevWeight

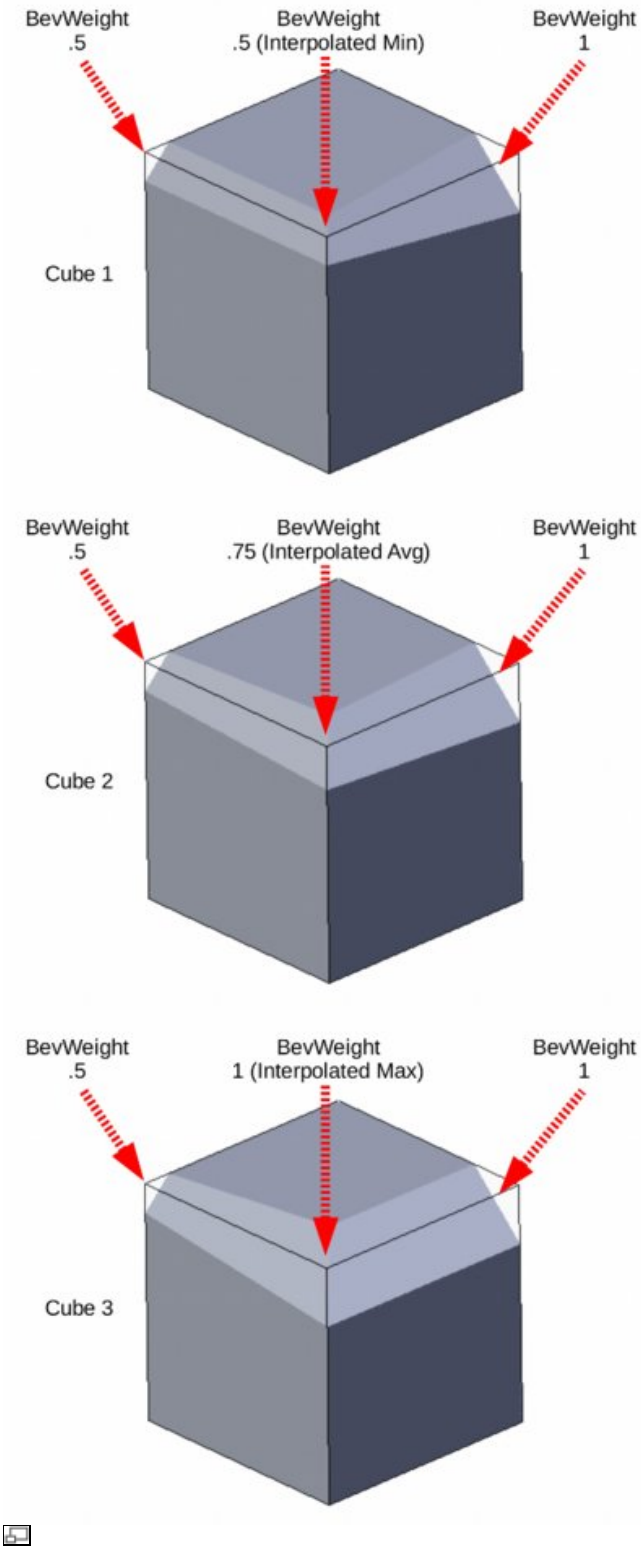
The BevWeight button in the Limit Using section when selected makes the Bevel Modifier take into account any Bevel Weights that may or may not be assigned to edges of the underlying mesh. When the BevWeight button is active and the edges of the underlying mesh actually have Bevel Weights assigned, the Bevel Modifier will use the values of the Bevel Weights to influence how much affect the Bevel Modifier will have on an edge (how much bevel that edge will get).



Bevel Modifier with BevWeight button active

When the BevelWeight button is active, 3 extra buttons appear underneath it named Min, Average and Max. These buttons control how the Bevel Weighted influences are calculated and assigned at places where 2 Bevel Weighted Edges meet that have different weights. Obviously when there are 2 Bevel Weighted edges which are directly linked/connected to each other, there needs to be a way to determine which particular Bevel Weight gets assigned at the point of contact between the 2 Bevel Weighted edges (Interpolation vertex point in the diagrams of the 3 cubes).

Here is an example screenshot showing the effect of different BevWeight Interpolation modes (Min, Avg & Max buttons):



Bevel Modifier with BevWeight Limit Using Min, Avg and Max interpolation mode buttons

- Limit Using: Min

    When 2 Bevel Weighted edges with different Bevel Weights meet; the Interpolation/ Contact point between the 2 Bevel Weighted edges will use the smaller of the 2 Bevel Weights as the Bevel Weight for the Interpolation/Contact point (See Cube 1).

- Limit Using: Average

When 2 Bevel Weighted edges meet; the Interpolation/Contact point between the 2 Bevel Weighted edges will use the Average weight of the 2 Bevel Weights as the Bevel Weight for the Interpolation/Contact point (See Cube 2).

- Limit Using: Max

  When 2 Bevel Weighted edges meet; the Interpolation/Contact point between the 2 Bevel Weighted edges will use the Larger weight of the 2 Bevel Weights as the Bevel Weight for the Interpolation/Contact point (See Cube 3).

Note

Thanks to Xalt for making clear how the Limit Using BevWeight Min, Avg and Max buttons affect underlying meshs. −− Terrywallwork −− 28th June 2008.

Examples

Tutorials and examples of how to use the modifier here.

See Also

Links to other related or useful places for further information here.

Previous: Manual/Array Modifier	Contents	*Next:* Manual/Booleans

User Manual: Contents | Guidelines | Blender Version **2.4x**

Boolean Modifier

Mode: Object Mode (Mesh objects only)

Panel: Editing Context Modifiers

Hotkey: W

Menu: Object Boolean Operation...

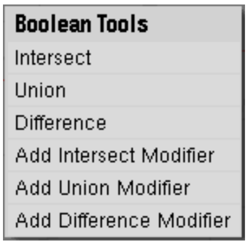
Description

Boolean operations are a method of combining or subtracting solid objects from each other to create a new form. Boolean operations in Blender only work on two Mesh type objects, preferably ones that are solid, or closed, with a well defined interior and exterior surface. If more than two mesh objects are selected only the *active* and previously selected object are used as operands. The boolean operations also take Materials and UV−Textures into account, producing objects with Material indices or multi UV−mapped objects.

Options

Boolean Menu

Using the Boolean menu (W in Object Mode) presents the following options:



Boolean operations

- Intersect

  Creates a new object whose surface encloses the volume *common to both* original objects.

- Union

  Creates a new object whose surface encloses the *total volume of both* original objects.

- Difference

  The only operation in which the order of selection is important, the active object is subtracted from the selected object. That is, the resulting object surface encloses a volume which is the volume belonging to the selected *and inactive* object, but *not* to the selected *and active* one.

- Add Intersect Modifier

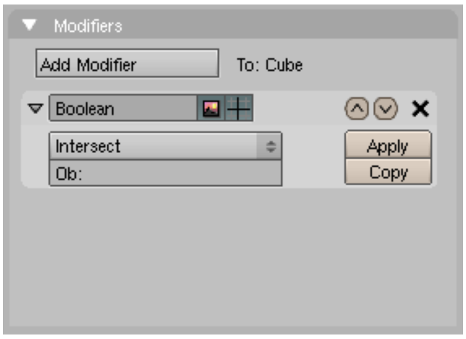
  A shortcut that applies a Boolean Modifier and selects *Intersect* in one step.
- Add Union Modifier

  A shortcut that applies a Boolean Modifier and selects *Union* in one step.
- Add Difference Modifier

  A shortcut that applies a Boolean Modifier and selects *Difference* in one step.

Boolean Modifier

This sub−panel appears in the Editing Context panel group which is accessed using F9 or clicking button in the Buttons window. This sub−panel is part of the Modifier parent panel. For further information about the common panel components see the Interface section on modifiers.



Modifier panel with Boolean modifier activated.

Intersect
: The available boolean operation types (Intersect/Union/Difference)

Ob
: The name of the object to be used as the second operand to this modifier.

The downside of using the direct Boolean commands is that in order to change the intersection, or even apply a different operation, you need to remove the new object and redo the command. Alternatively, one can use a Boolean Modifier for great flexibility and non−destructive editing. As a modifier the booleans can be enabled/disabled or even the order rearranged in the stack. In addition, you can move the operands and see the boolean operation applied interactively in real time!

Caution

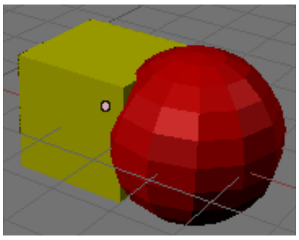
if the objects' meshes are too complex you may be waiting a while as the system catches up with all the mouse movements. Turning off display in the 3D View in the modifier panel can improve performance.

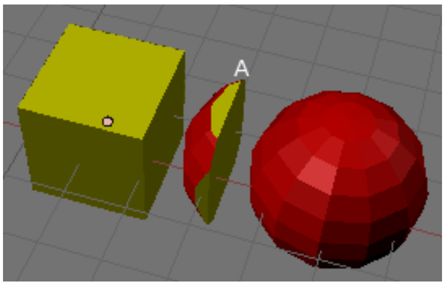
To get the final object created (as with the direct Boolean tools) you need to "Apply" the modifier using the modifier's Apply button, and to see the new object you need to move the remaining operand away or switch to local view NumPad /. Until you apply the modifier, a new mesh object will not be created. When you apply the boolean modifier you are notified that any mesh sticky information, animation keys and vertex information will be deleted.

Examples

Intersect

The cube and the sphere have been moved to reveal the newly created object (**A**). Each face of the new object has the material properties of the corresponding surface that contributed to the new volume based on the Intersect operation.

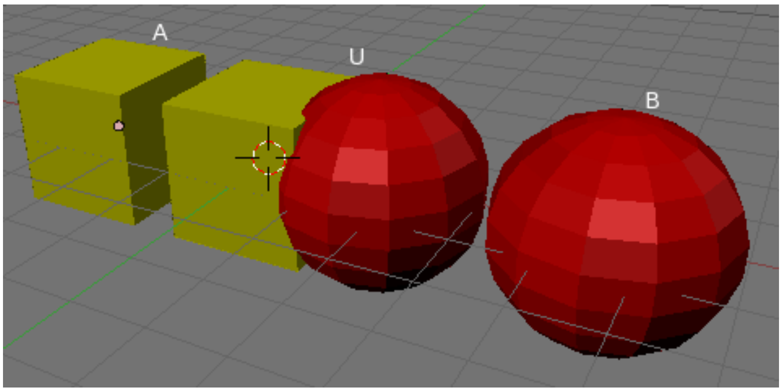

Intersect Before



Intersect After

Union

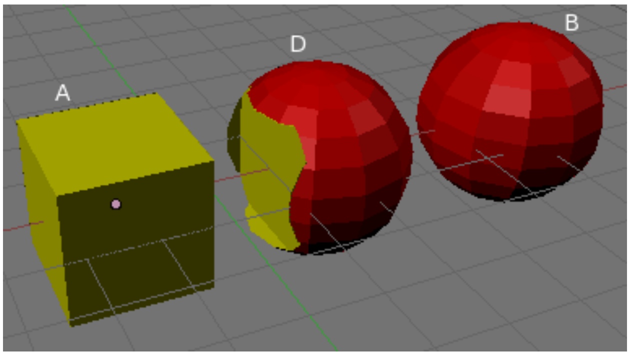
The cube (**A**) and the sphere (**B**) have been moved to reveal the newly created object (**U**). **U** is now a single mesh object and the faces of the new object have the material properties of the corresponding surface that contributed to the new volume based on the Union operation.



Union example

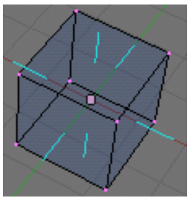
Difference

The Difference of two objects is not commutative in that the *active* object minus the *inactive* object does not produce the same as *inactive* minus *active*. The cube (**A**) has been subtracted from a sphere (**B**), and both have been moved to reveal the newly created object (**D**). **D** is now a single mesh object and the faces of the new object have the material properties of the corresponding surface that contributed to the new volume based on the Difference operation. **D**'s volume is less than **B**'s volume because it was decreased by subtracting part of the cube's volume.



Difference example

Technical Details



Visible normals

The boolean operations rely heavily on the surface normals of each object and so it is very important that the normals are defined properly and consistently. This means each object's normals should point outward. A good way to see the object's normals is to turn on the visibility of normals using the Mesh Tool 1 panel; the panel is accessable from the *Buttons window*, using F9 and clicking *Draw normals*. The normals are only visible while in Edit mode. (*Visible normals*) is an example of a cube with its normals visible. See *The Interface* for a full description of the Mesh Tool 1 panel.

In the case of *open* objects, that is objects with holes in the surface, the interior is defined mathematically by extending the boundary faces of the object to infinity. As such, you may find that you get unexpected results for these objects. A boolean operation never affects the original objects, the result is always a new object.

Warning

This is NOT TRUE with the modifiers Boolean: when they are applied, they modify their owner object, and do not create a new one!

Some operations will require you to move the operands or switch to local view NumPad / to see the results of the boolean operation.

Limitations & Work−arounds

The number of polygons generated can be very large compared to the original meshes, especially when using complex concave objects. Furthermore, the polygons that are generated can be of poor quality, for example, very long and thin and sometimes very small. Try using the Decimate Modifier (EditButtons F9) to fix this problem.

Sometimes the boolean operation can fail with a message saying ("An internal error occurred −− sorry"). If this occurs, try to move or rotate the objects just a very small amount and try again.

See Also

- Modifiers: Booleans

Previous: Manual/Bevel Modifier	Contents	*Next:* Manual/Build Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

Build Modifier

Mode: Object Mode

Panel: Editing Context Modifiers

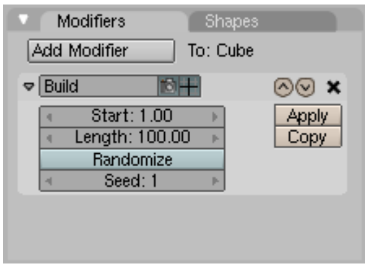
Hotkey: F9

Description

The Build modifier causes the faces of the Object to appear, one after the other, over time. If the Material of the Mesh is a Halo Material, rather than a standard one, then the vertices of the Mesh, not the faces, appear one after another.

Faces or vertices appear in the order in which they are stored in memory. This order can be altered by selecting the Object and pressing Ctrl F out of EditMode to sort the face based on their local Z axis height.

Options



Build modifier

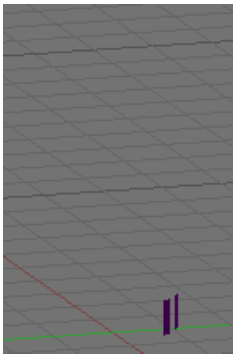
Start
: The start frame of the building process

Length
: The number of frames over which to build up from 0% to 100%

Randomize
: Randomizes the order that the faces are built up

Seed
: The random seed. Change this to get a different random (but deterministic) effect

Example



Imagine a city being built right before your eyes. How about a fountain of halos? This example shows a simple tube, beauty short subdivided a few times, with the modifier turned on, from frames 1 to 100 (in steps of 10) with no additional animation.

Previous: Manual/Booleans	Contents	*Next:* Manual/Cast Modifier

User Manual: Contents | Guidelines | Blender Version **2.44**

Cast Modifier

Mode: Any Mode

Panel: Editing Context Modifiers

Hotkey: F9

Description

This modifier shifts the shape of a mesh, curve, surface or lattice to any of a few pre−defined shapes (sphere, cylinder, cuboid).

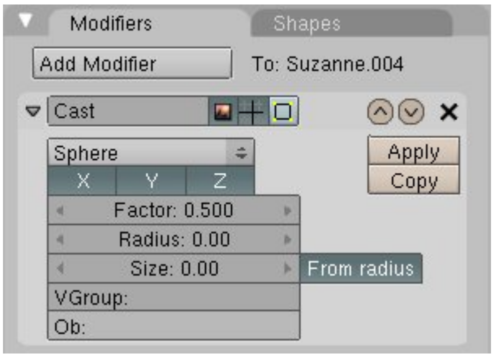
It is equivalent to the To Sphere button in the Editing Context Mesh Tools panel and the To Sphere command in the Editing Context Mesh Transform menu, as well as what other programs call "Spherify" or "Spherize", but, as written above, it is not limited to casting to a sphere.

Hint: the Smooth modifier is a good companion to Cast, since the casted shape sometimes needs smoothing to look nicer or even to fix shading artifacts.

Important

For performance, this modifier works only with local coordinates. If the modified object looks wrong, you may need to apply the object's rotation (hotkey: CTRL+a), specially when casting to a cylinder.

Options



Cast modifier

Type
: Menu to choose cast type (target shape): Sphere, Cylinder or Cuboid.

X Y Z
: Toggle buttons to enable / disable the modifier in the X, Y, Z axes directions.

Factor
: The factor to control blending between original and casted vertex positions. It's a linear interpolation:

0.0 gives original coordinates (aka modifier has no effect), 1.0 casts to the target shape. Values below or above [0.0, 1.0] deform the mesh, sometimes in interesting ways.

Radius
: If nonzero, this radius defines a sphere of influence. Vertices outside it are not affected by the modifier.

Size
: Alternative size for the projected shape. If zero, it is defined by the initial shape and the control object, if any.

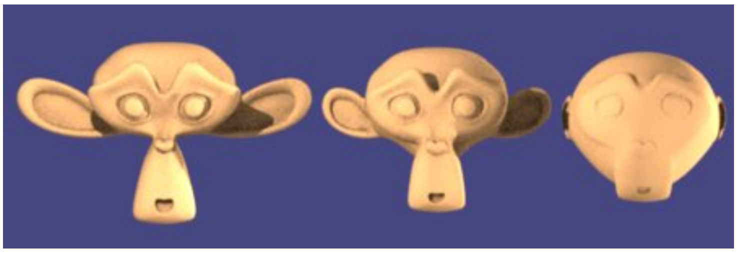
From radius
: Toggle: if ON, calculate Size from Radius, for smoother results.

Object
: The name of an object to control the effect. The location of this object's center defines the center of the projection. Also, its size and rotation transform the projected vertices. Hint: animating (keyframing) this control object also animates the modified object.

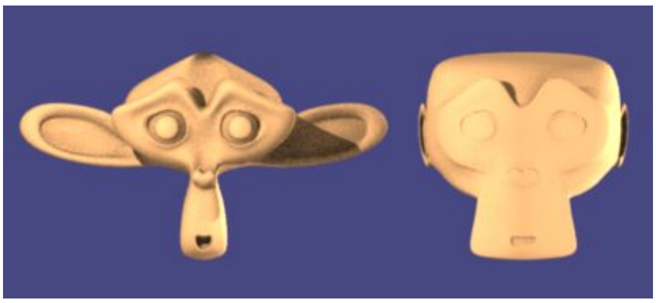
VGroup
: A vertex group name, to restrict the effect to the vertices in it only. This allows for selective, realtime casting, by painting vertex weights.

Examples

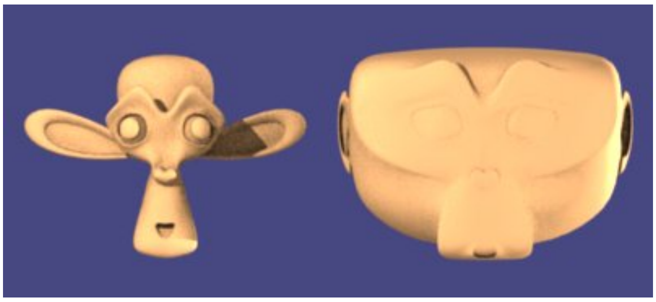
These models also have Subsurf modifiers applied after the Cast modifier:



Casting to sphere: **Left**: Factor: −0.8. **Center**: Factor: 0.0 (disabled). **Right**: Factor: 0.8.

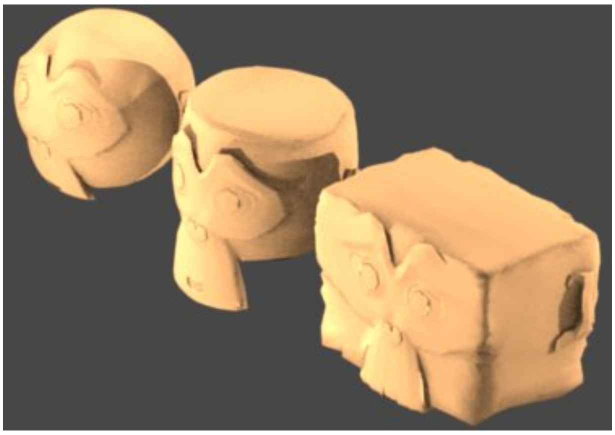


Casting to cylinder: **Left**: Factor: −0.8. **Right**: Factor: 0.8.



Casting to cuboid: **Left**: Factor: −0.8. **Right**: Factor: 0.8.

These have Subsurf (in simple subdiv mode) applied before the Cast modifier, to have more vertices to work with:



Casting Suzanne: **Left**: to sphere; **Center**: to cylinder. **Right**: to cuboid.

Previous: Manual/Build Modifier	Contents	*Next:* Manual/Cloth Modifier

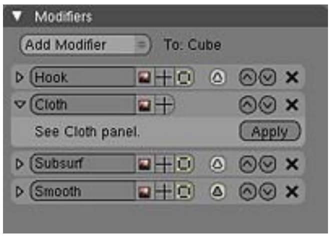
User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.46**

Cloth and the Modifier Stack

Mode: Object Mode

Panel: Editing Context â†’ Modifiers

Hotkey: F9



Cloth is a mesh modifier, in that it can modify the shape of the mesh. It is inserted in the modifier stack when you designate a mesh as Cloth. When used with other modifiers, you can achieve very convincing results. For more information on using Cloth, see this page.

Modifier order is important

- Any subsurf or smoothing modifiers should belong BELOW the cloth modifier
- Any armature, hook or other deforming modifiers should go ABOVE the cloth modifier.
- The cloth modifier will only take the deformations of pinned/weight painted vertices into account! That means that if you like to have e.g. a hook modifier to move a vertex you need to enable "pin" in the first cloth tab and paint that hooked vertex red for full influence of the hook modifier.

If you Apply the Cloth modifier, you remove the simulation but leave the mesh intact and deformed according to that frame of the simulation.

If you delete the modifier, the mesh goes back to its original (starting) shape and isn't a cloth simulation anymore.

Previous: Manual/Cast Modifier	Contents	*Next:* Manual/Curve Modifier

User Manual: Contents | Guidelines | Blender Version **2.46**

Curve Modifier

Mode: Object Mode

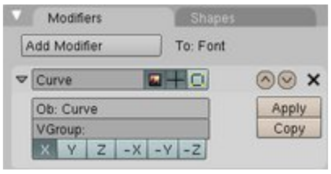
Panel: Editing Context â†’ Modifiers

Hotkey: F9

Description

The Curve modifier functions just like its predecessor with the added exception that there is no need for a parent/child relationship between the curve and the object being deformed, and that the effect can be applied to all object types in realtime.

Options



Curve modifier panel.

Ob
: The name of the curve object that will affect the deforming object.

VGroup
: Vertex Group name. Name of vertex group within the deforming object. The modifier will only affect vertices assigned to this group.

X, Y, Z, −X, −Y, −Z
: This is the axis that the curve deform along.

Example

Curve modifier example.

This example shows a mesh, the film base, that is deformed by the Curve modifier. The curve is the single black line running through the wheels. To animate the film going through the wheels is just a matter of dragging the film mesh along the curve axis!

Previous: Manual/Cloth Modifier | Contents | *Next:* Manual/Decimate Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

Decimate Modifier

Mode: Object Mode

Panel: Editing Context Modifiers

Hotkey: F9

This sub−panel appears in the Editing Context panel group which is accessed using F9 or clicking button in the Buttons window. This sub−panel is part of the Modifier parent panel. For further information about the common panel components see the Modifier Stack section.

Description

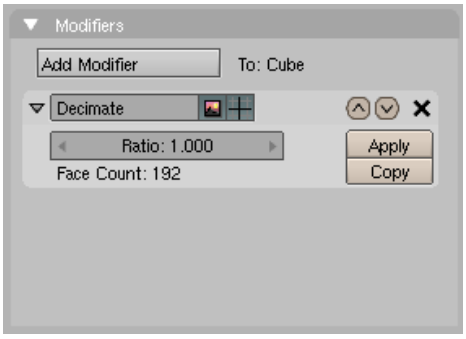
The Decimate modifier allows you to reduce the vertex/face count of a mesh with minimal shape changes. This is not applicable to meshes which have been created by modeling carefully and economically, where all vertices and faces are necessary to correctly define the shape, but if the mesh is the result of complex modeling, with proportional editing, successive refinements, possibly some conversions from SubSurfed to non−SubSurfed meshes, you might very well end up with meshes where lots of vertices are not really necessary.

The Decimate Modifier is a quick and easy way of reducing the polygon count of a mesh non−destructively. This modifier demonstrates of the advantages of a mesh modifier system because it shows how an operation, which is normally permanent and destroys original mesh data, can be done interactively and safely using a modifier.

Unlike the majority of existing modifiers, the Decimate Modifier does not allow you to visualize your changes in edit mode.

The Decimator tool only handles triangles, so each quadrilateral face is implicitly split into two triangles for decimation.

Options



Decimate modifier

Ratio

The percentage of faces to keep after decimation, from **0.0** (0%, all faces have been completely removed) to **1.0** (100%, mesh is completely intact, except quads have been triangulated).
As the percentage drops from **1.0** to **0.0** the mesh becomes more and more decimated until the mesh no longer visually looks like the original mesh.

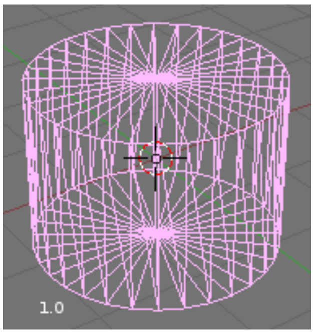
Face Count: (Display only)
This field shows the number faces as a result of applying the Decimate Modifier.

Examples

Simple plane

A simple example is a plane, and a 4x4 undeformed Grid object. Both render exactly the same, but the plane has 1 face and 4 vertices, while the grid has 9 faces and 16 vertices, hence lots of unneeded vertices and faces. The Decimate Modifier allows you to eliminate these unneeded faces.

Decimated cylinder

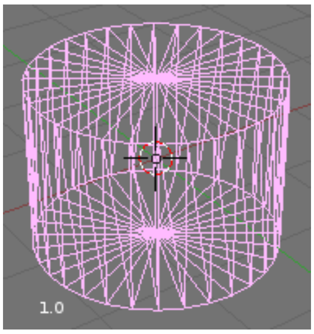


Decimate applied

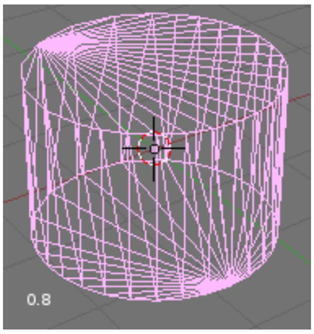
We take a simple example of decimating a cyclinder using the default of 32 segments. This will generate a cyclinder with 96 faces. When the Decimate Modifier is applied (*Decimate applied*) the face count goes up! This is because the Modifier converts all quadrilaterals into triangles (*tris*) which always increases the face count. Each quadrilateral (*quads*) decomposes into two triangles.

The main purpose of the Decimate Modifier is to reduce mesh resources through a reduction of vertices and faces, but at the same time maintain the original shape of the object.

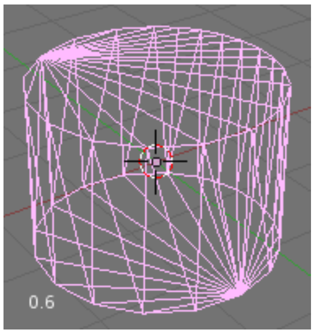
Each of the following images has had the percentage dropped in each successive image, from 1.0 to 0.05. Notice that the face count has gone from 128 to 88 at 0.6% and yet the cyclinder continues to look very much like a cylinder and we discarded 40 unneeded faces.



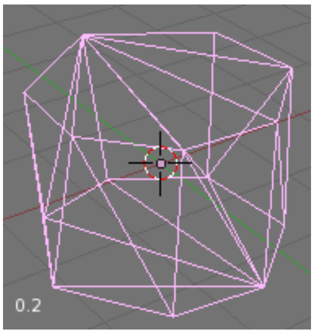
1.0%. Face count (128)



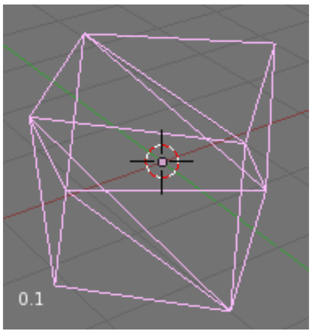
0.8%. Face count (102)



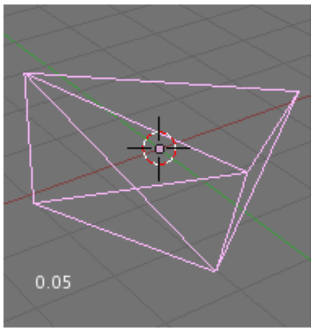
0.6%. Face count (88)



0.2%. Face count (24)



0.1%. Face count (12)



0.05%. Face count (6)

As you can see when the percentage has reached 0.1 the cylinder looks more like a cube. And when it has reached 0.05 it doesn't even look like a cube!

Once you have reached the face count and appearance you were looking for you can "Apply" the modifier. If you want to convert as many of the tris back to quads to reduce mesh resources further you can use (Alt J) while in Edit mode.

High resolution landscape

Decimated landscape, top: original; middle: lightly decimated; bottom: heavily decimated.

Decimated landscape, top: original; middle: lightly decimated; bottom: heavily decimated. shows a landscape generated via a careful application of the Noise technique described earlier, on a quite vast grid. On top, the result for the original mesh and below, two different levels of decimation. To the eye the difference is indeed almost unnoticeable, but as the vertex count goes down there is a huge gain.

Previous: [Manual/Curve Modifier]	[Contents]	*Next:* [Manual/Displace Modifier]

User Manual: Contents | Guidelines | Blender Version **2.45**

Displace Modifier

Mode: Any Mode

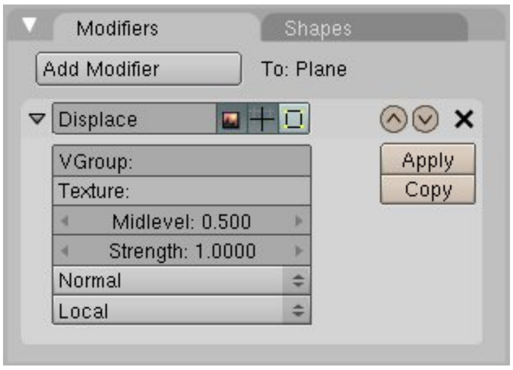
Panel: Editing Context Modifiers

Hotkey: F9

Description

The Displace modifier displaces vertices in a mesh based on the intensity of a texture. Either procedural or image textures can be used. The displacement can be along a particular local axis, along the vertex normal, or the separate RGB components of the texture can be used to displace vertices in the local X, Y and Z directions simultaneously.

Options



Displace modifier

VGroup
: The name of a vertex group which is used to control the influence of the modifier.
  ◊ If VGroup is empty, the modifier affects all vertices equally.

Texture
: The name of the texture datablock from which the displacement for each vertex is derived.
  ◊ If this field is empty, the modifier will be disabled.

Midlevel
: The texture value which will be treated as no displacement by the modifier. Texture values below this value will result in negative displacement along the selected direction, while texture values above this value will result in positive displacement. This is achieved by the equation (displacement) = (texture value) − Midlevel.

Strength
: The strength of the displacement. After offsetting by the Midlevel value, the displacement will be multiplied by the Strength value to give the final vertex offset. This is achieved by the equation (vertex offset) = (displacement) * Strength.

◊ A negative strength can be used to invert the effect of the modifier.

Direction

The direction along which to displace the vertices.
Can be one of the following:

◊ X − displace along local X axis
◊ Y − displace along local Y axis
◊ Z − displace along local Z axis
◊ RGB −> XYZ − displace along local XYZ axes individually using the RGB components of the texture
◊ Normal − displace along vertex normal

Texture Coordinates

The texture coordinate system to use when retrieving values from the texture for each vertex.
Can be one of the following:

◊ UV − take texture coordinates from face UV coordinates; if the object has no UV coordinates, use the Local coordinate system

Note

Since UV coordinates are specified per face, the UV texture coordinate system currently determines the UV coordinate for each vertex from the first face encountered which uses that vertex; any other faces using that vertex are ignored. This may lead to artifacts if the mesh has non−contiguous UV coordinates.

◊ Object − take the texture coordinates from another object's coordinate system (specified by the Ob field)
◊ Global − take the texture coordinates from the global coordinate system
◊ Local − take the texture coordinates from the object's local coordinate system

Ob

The object from which to take texture coordinates.

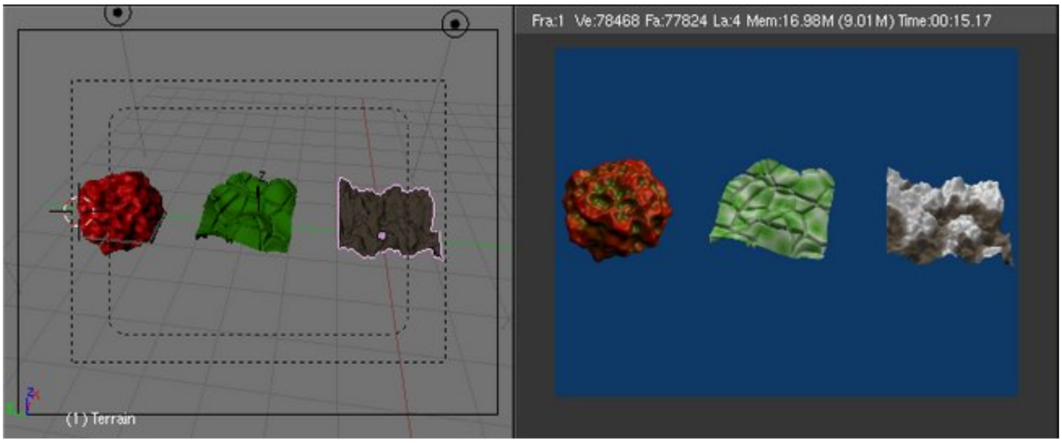
◊ This field is only visible when the Object texture coordinate system is selected.
◊ If this field is blank, the Local coordinate system is used
◊ Moving the object will therefore alter the coordinates of the texture mapping.

UV Layer

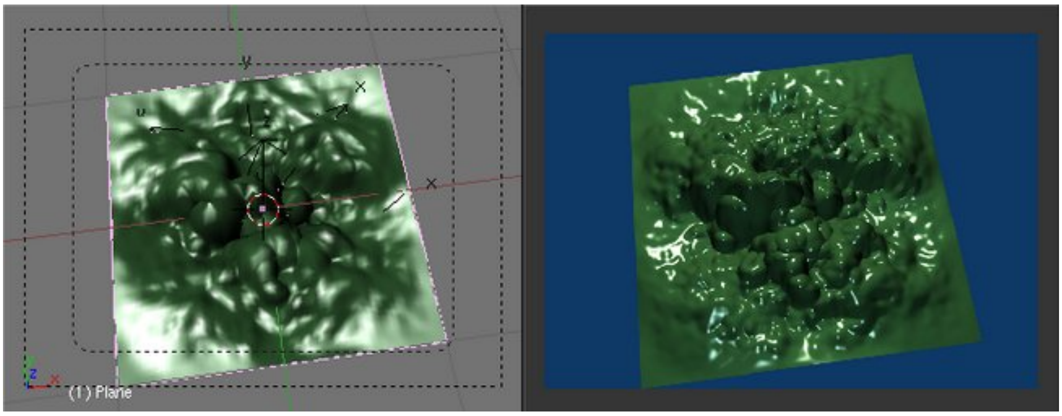
The UV coordinate layer from which to take texture coordinates.

◊ This field is only visible when the UV texture coordinate system is selected.
◊ If this field is blank, but there is a UV coordinate layer available (e.g. just after adding the first UV layer to the mesh), it will be overwritten with the currently active UV layer.

Examples



Three different objects created with the Displace modifier.
Sample blend file



A slime animation created with the Displace modifier.
YouTube video
Sample blend file

Previous: Manual/Decimate Modifier Contents *Next:* Manual/EdgeSplit Modifier

User Manual: Contents | Guidelines | Blender Version **2.43**

EdgeSplit Modifier

Mode: Any Mode

Panel: Editing Context Modifiers

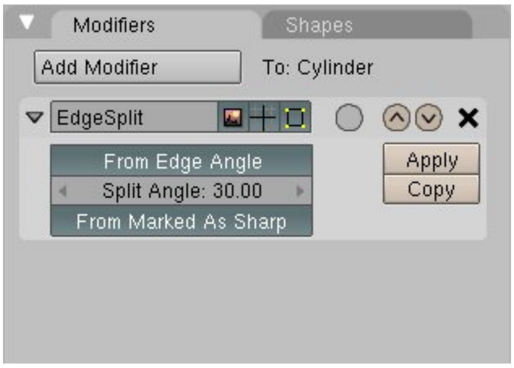
Hotkey: F9

Description

The EdgeSplit modifier splits edges within a mesh. The edges to split can be determined from the edge angle or edges marked as sharp can be split.

Splitting an edge affects vertex normal generation at that edge, making the edge appear sharp. Hence, this modifier can be used to achieve the same effect as the Autosmooth button, making edges appear sharp when their angle is above a certain threshold. It can also be used for manual control of the smoothing process, where the user defines which edges should appear smooth or sharp (see Mesh Smoothing for other ways to do this). If desired, both modes can be active at once. The output of the EdgeSplit modifier is available to export scripts, making it quite useful for creators of game content.

Options



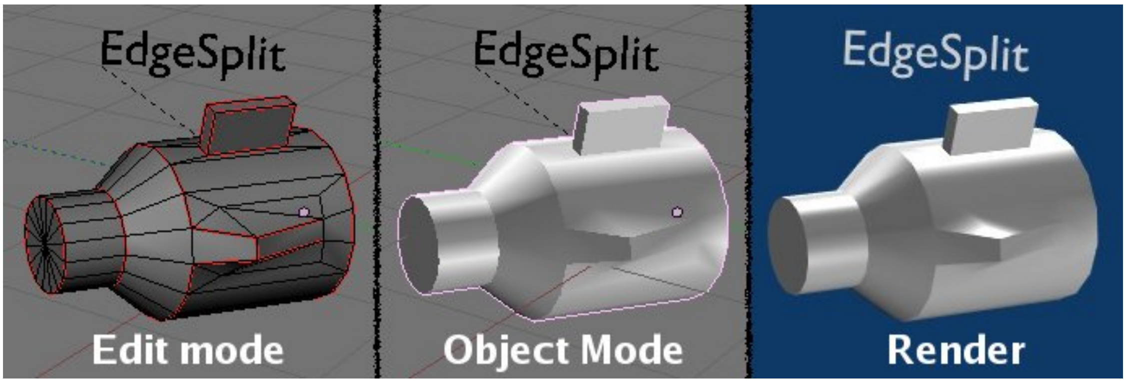
EdgeSplit modifier

From Edge Angle
: If this button is enabled, edges will be split if their edge angle is greater than the Split Angle setting.
  - ◊ The edge angle is the angle between the two faces which use that edge.
  - ◊ If more than two faces use an edge, it is always split.
  - ◊ If fewer than two faces use an edge, it is never split.

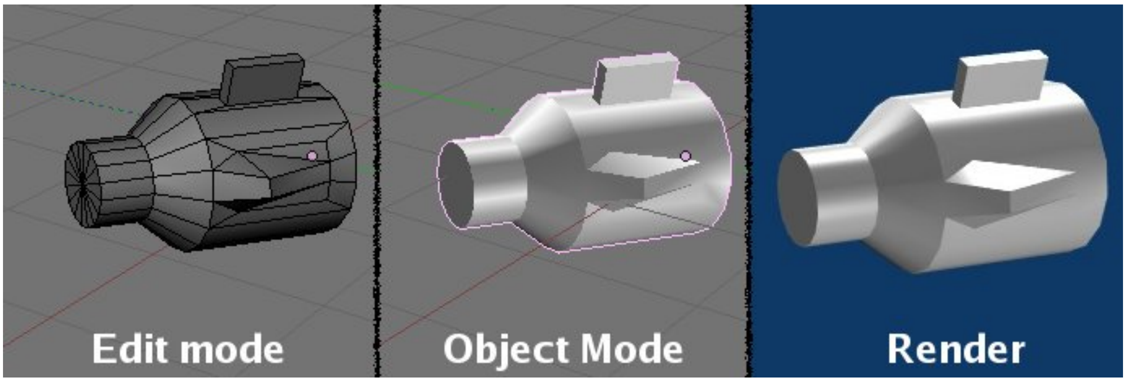
Split Angle
: This is the angle above which edges will be split if the From Edge Angle button is selected, from 0° (all edges are split) to 180° (no edges are split).

From Marked As Sharp
: If this button is enabled, edges will be split if they are marked as sharp, using the Edge Specials Mark Sharp menu item, accessible by (Ctrl E) in editmode.

Examples



EdgeSplit modifier output with From Marked As Sharp selected.
Sample blend file



EdgeSplit modifier output with From Edge Angle selected.
Sample blend file

Previous: Manual/Displace_Modifier Contents *Next:* Manual/Explode Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

Explode Modifier

Mode: Any Mode

Panel: Editing Context Modifiers

Hotkey: F9

Description

The Explode Modifier is used to alter the mesh geometry (by moving/rotating its faces) in a way that (roughly) tracks underlying emitted particles that makes it look as if the mesh is being exploded (broken apart and pushed outward).

For the Explode Modifier to have a visible effect on the underlying mesh it has to be applied to a mesh which has a particle system on it, in otherwords it has to be a mesh which outputs particles. This is because the particle system on the mesh is what controls how a mesh will be exploded, and therefore without the particle system the mesh wont appear to alter. Also both the number of emitted particles and number of faces determine how granular the Explode Modifier will be. With default settings the more faces and particles the more detailed the mesh exploding will be, because there are more faces and particles to affect detachment/movement of faces.

Here is a link to an Ogg Theora Movie showing a cube with a particle system and Explode Modifier applied:

Media:Manual − Explode Modifier − Exploding Cube.ogg

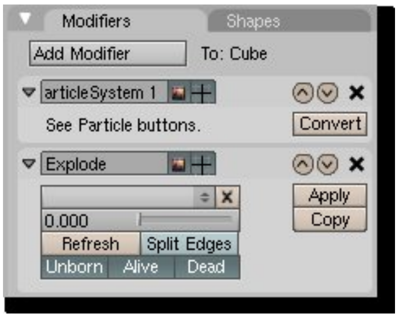
Here is a link to the original Blender file which has an Exploding cube setup, just free the particle cache by pressing the Free Bake button in the bake panel and then press the Animate button to see the animation:

Media:Manual − Explode Modifier − Exploding Cube.blend

Options

- Stacking Order Importance

  This modifier is highly affected by its position within the modifier stacking order. If it is applied before a Particle System modifier it will not be affected by particles and therefore appear to do nothing. The Particle System Modifier must appear before the Explode Modifier because the Particle System Modifier has the information needed to drive the Explode Modifier.



Explode Modifier panel with Particle System Modifier above it.

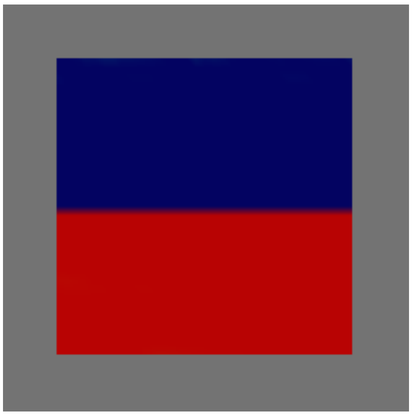
- Protect this vertex group

If a mesh that has an Explode Modifier on it also has vertex groups assigned to it then this field will allow the selection of one of those vertex groups. This will indicate to the Explode Modifier that it should take into account the weight values assigned to areas of the selected vertex group. Then depending on the weights assigned to that vertex group; either completely protect those faces from being affected by the Explode Modifier (which would happen if the faces had a weight value of 1) or completely remove protection from those faces (which would happen if the faces had a weight value 0).

Explode Modifier panel with "Protect this vertex group" field highlighted in yellow.

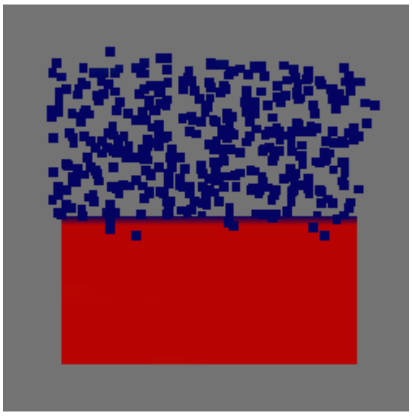
As well as completely protecting a face (with a weight painting of 1) or completely allowing a face to be exploded (weight painting of 0). It is possible to have in between weight paint values (weight painting values that arn't 0 or 1) that tell the Explode Modifier to only partially effect the underlying mesh.

Below is a plane which has been weight painted in 2 different weight paints, red which represents a weight paint of 1, and blue which represents a weight paint of 0. The reason that weight painting is used is because whenever you weight paint something it automatically makes a new vertex group (if the plane doesn't already have one), which can then be used with the "Protect this vertex group" field.


Weight painted plane before Explode Modifier is run.

Below is the same plane while the Explode Modifier is in the process of exploding the plane. As can be seen the weight painting means that the red (weight paint of 1) faces are left unmodified, and the blue (weight paint of 0) faces are modified.

Remember that by default weight painting makes a vertex group, that is what needs to be selected in the "Protect this vertex group" field for the weight painting to be taken notice of.



Weight painted plane after Explode Modifier is run. Showing the red area unmodified and the blue area exploded.

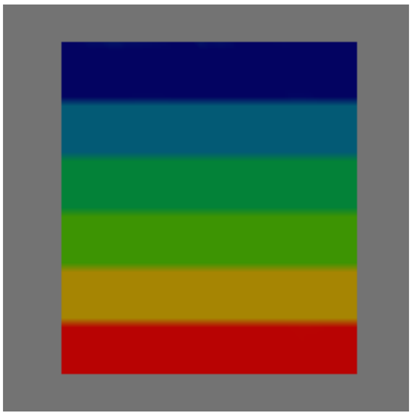
Here is a link to an Ogg Theora Movie showing a weight painted plane, painted with minimum and maximum weights to be animated with the Explode Modifier:

Media:Manual − Explode Modifier − Dual Weighted Plane.ogg

Here is a link to the original Blender file which the above movie was made from, just free the particle cache by pressing the Free Bake button in the bake panel and then press the Animate button to see the animation:

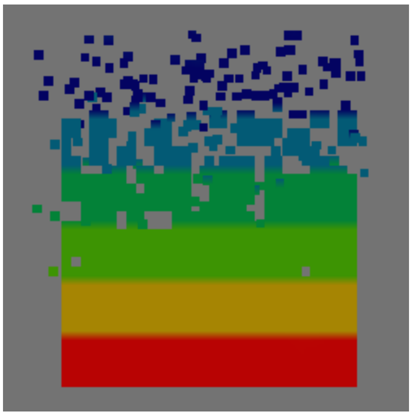
Media:Manual − Explode Modifier − Dual Weighted Plane.blend

Below is a plane which has got a blended weight paint on it, with RED being a weight paint of 1 through to other lower weighted colours all the way to BLUE which is a weight paint of 0.



Blended Weight painted plane before the Explode Modifier is run.

Below is the same blended weight painted plane after it has had the Explode Modifier running on it.



Blended weight painted plane after the Explode Modifier is run.

As can be seen the top part of the blended plane is highly affected by the Explode Modifier; whereas, lower down the plane, where the weight painting gets higher in weight, is affected less and less by the Explode Modifier.

Here is a link to an Ogg Theora movie showing a blended weight painted plane, painted with multiple weights animated with the Explode Modifier:

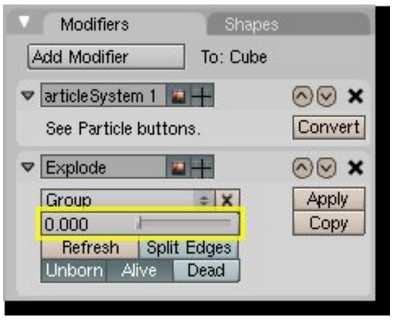
[Media:Explode_Modifier_−_Graded_Weighted_Plane_−_Exploded.ogg]

Here is a link to the original Blender file which the above movie was made from, just free the particle cache by pressing the Free Bake button in the bake panel and then press the Animate button to see the animation:

[Media:Explode_Modifier_−_Graded_Weighted_Plane.blend]

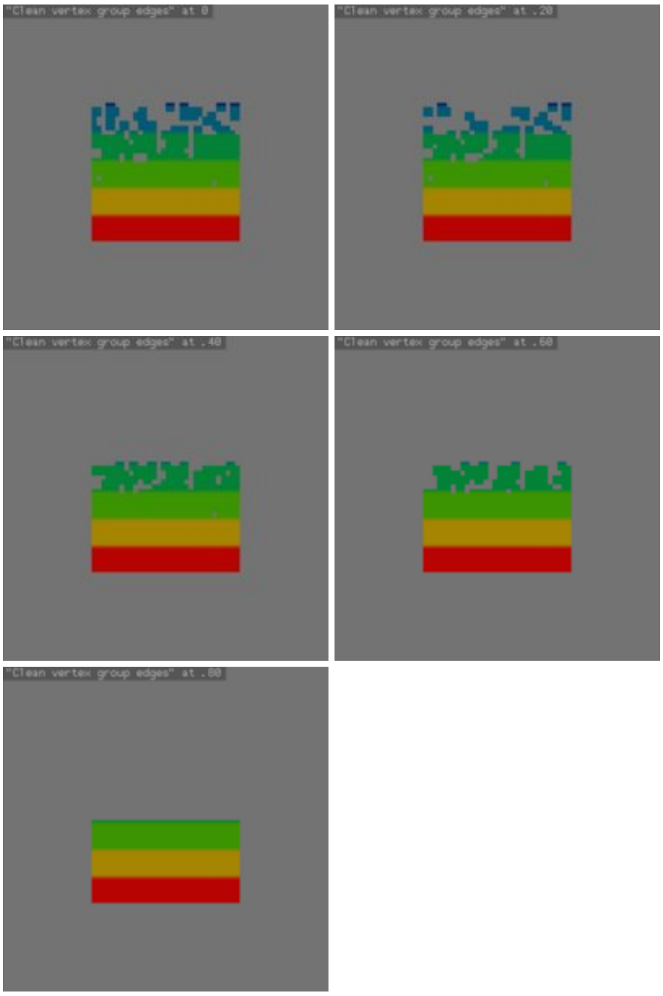
- Clean vertex group edges

This numeric slider field appears to erode the exploded edges around a vertex groups edges (making edges less jaggied). The slider can have a value between 0 (no eroding of edges) and 1 maximum erosion of edges.



The Clean vertex group edges numeric slider field highlighted in yellow.

Below are some screenshots of a mesh that has had an Explode Modifier applied to it with various values for "Clean vertex group edges":

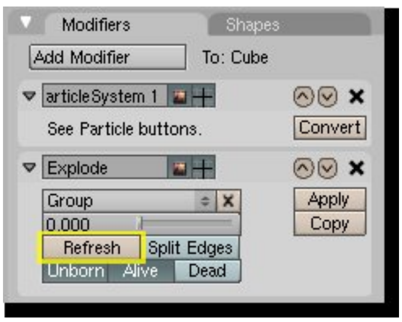


Caution

I have been unable to find out a definitive explanation of exactly how the "Clean vertex group edges" works on a mesh and what the slider value represents, so I have just used my best guess from looking at the result of using it and setting it with different values. If you have a more concrete definition of what is does please alter this text to reflect it or contact me and I will update the text. −−Terrywallwork

- Refresh (Recalculate faces assigned to particles)

  When certain changes are made to the underlying Explode Modifier mesh (such as changing the vertex group weightings of faces) the faces displayed and the particles that influence certain faces can get out of sync. If this happens pressing the Refresh button will tell the Explode Modifier to update all it's calculations to take into account new settings.



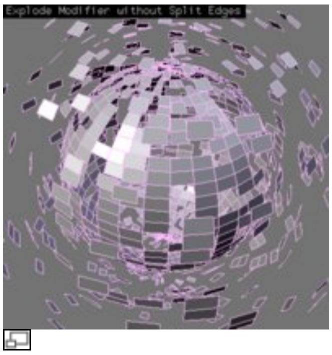
Explode Modifier Panel with the Refresh button highlighted in yellow.

As an example, if you have a mesh that has faces that get modified by the end of an Explode Modifier run and you then alter those faces vertex group weights so they have a value of 1 (meaning they can't get altered in future (when the "Protect this vertex group" option is active)) the Explode Modifier will still show the old altered state of the faces until you click the Refresh button.

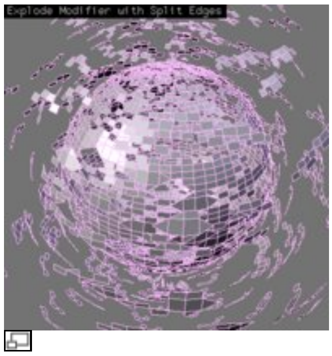
- Split Edges

When this button is selected faces which are exploded are split making them smaller and more fragmented, giving more faces for the Explode Modifier to work with and usually giving better explode results.

Below are 2 screenshots of a Explode Modifier in progress; one with no Split Edges option active and one with Split Edges activated:



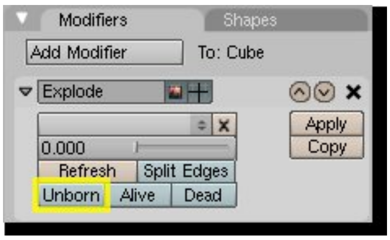
Uvsphere without Split Edges activated on the Explode Modifier.



Uvsphere with Split Edges activated on the Explode Modifier.

- Unborn

This button controls whether mesh faces will be visible or not before a particle for it has been created/born.

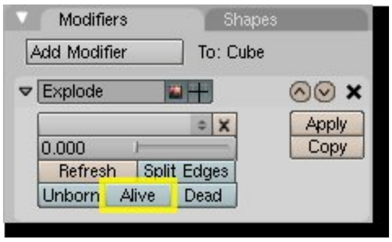


Explode Modifier Panel with the Unborn button highlighted in yellow.

What this means for example, is if you have an Explode Modifier on a mesh and certain faces on that mesh do not currently have any created/born particles on them then those faces will not be visible if the Unborn button is not selected, making faces appear to pop into visibility when particles are created/born.

- Alive

This button controls whether mesh faces will be visible or not while a particle for it is Alive/Active.

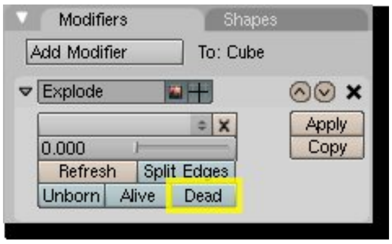


Explode Modifier Panel with the Alive button highlighted in yellow.

What this means for example, is if you have an Explode Modifier on a mesh and certain faces on that mesh have particles that are Alive/Active on them then those faces will be visible if the Alive button is selected.

- Dead

This button controls whether mesh faces will be visible or not when the particle associated with the face is dead.



Explode Modifier Panel with the Dead button highlighted in yellow.

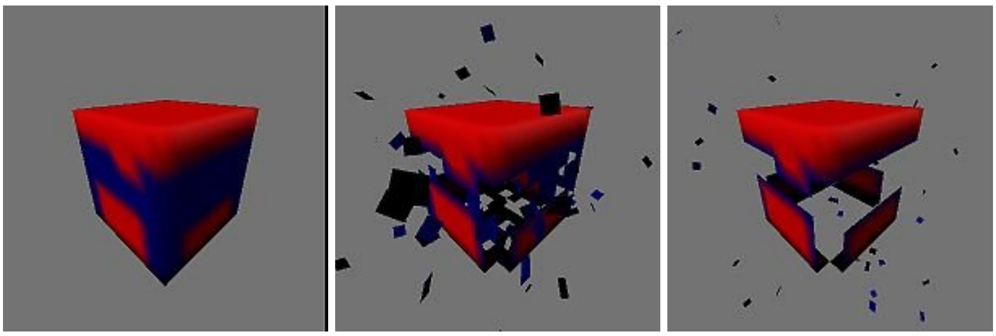
What this means for example, is if you have an Explode Modifier on a mesh and certain faces on that mesh have particles that are Dead on them then those faces will be visible if the Dead button is selected.

Examples

Exploding Cube

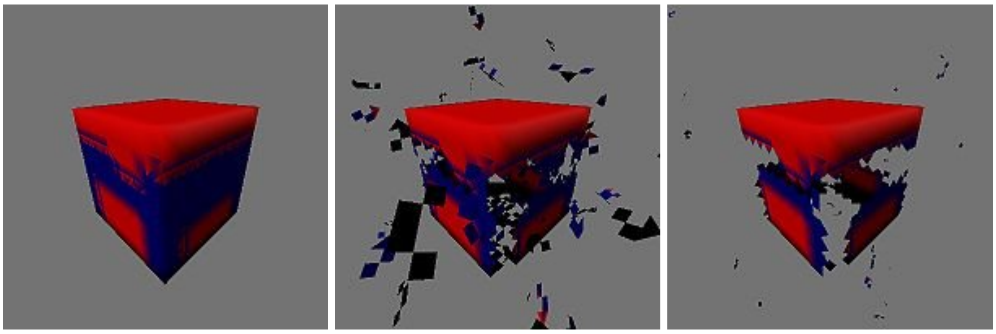
Weight painted object with *SplitEdge* option not selected

Note that in the image to the right, the faces are not jagged.



Weight painted object with *SplitEdge* option selected

Note that in the image to the right, the edges are jagged



Download .blend file

Media:Manual − Explode Modifier − Example Exploding Cube.blendâ€

See Also

Using the Explode Modifier with a Particle System to break meshes apart

Previous: Manual/EdgeSplit Modifier	Contents	*Next:* Manual/Hooks

User Manual: Contents | Guidelines | Blender Version **2.4x**

Hooks are similar to Shape Keys in that they deform a mesh over time (frames). The difference is that hooks make it look like the mesh is snagged with a fish hook. Moving the hook moves selected vertices under the influence of the hook (which is really just an Empty), and you make the hook move by animating the motion of the empty through Ipo keys. As the hook moves, it pulls weighted vertices from the mesh with it. If you have used Proportional Editing, you can think of it as animated proportional editing. While hooks do not give you the fine control over vertice movement that Shape Keys do, they are much simpler to use.

Object Hooks

Mode: Object Mode / Edit Mode

Panel: Editing Context Modifiers

Hotkey: Ctrl H

Menu: Mesh Vertices Add Hook

Description

Hooks give access at object level to the underlying geometry of meshes, curves, surfaces or lattices. A hook is an object feature and it is like a parent to an object, but for vertices. You can create as many hooks to an object as you like, and assign for each hook vertices that will be affected. Overlapping hooks is also possible, here a weighting factor per hook is provided that determines the amount each hook will affect the overlapping vertices.

All object level options and transformations are possible on the hook object, including using hierarchies, constraints, ipo and path animations. You can also make the hook−parent a child of the original object if you don't want object transformations to deform the hooks.

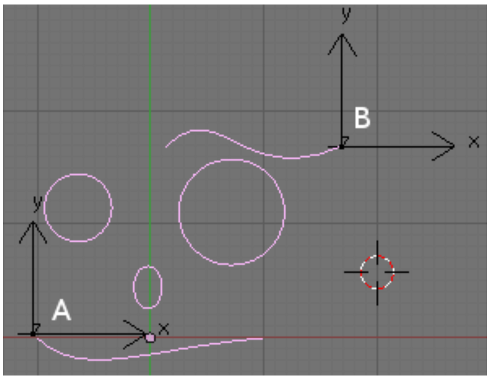
Note

When you change topology (i.e. more destructive editing than just manipulating existing points), you most likely have to reassign existing hooks as well.

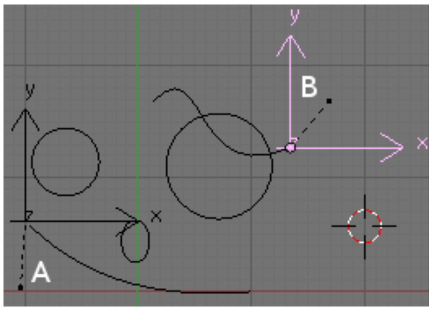
Examples

A typical example of using Hooks is to animate vertices or groups of vertices. For example, you may want to animate the vertices associated with a "Mouth" on a character's face.

In (*Animated face frame 1*) and (*Animated face frame 10*) a face made from Bezier curves has two Hooks applied. One applied to a control−point on the mouth labeled "**A**" and one applied to the eyebrow labeled "**B**". The animation has 10 frames over which Hook **A** moves up and Hook **B** moves down.



Animated face frame 1.



Animated face frame 10.

Adding Hooks

Mode: Edit Mode

Panel: Editing Context Modifiers

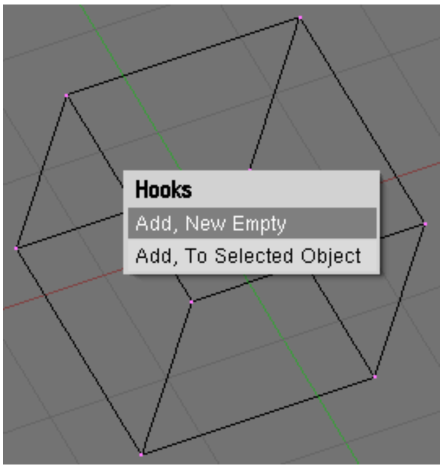
Hotkey: Ctrl H

Menu: Mesh Vertices Add Hook

Description

Since hooks relate to vertices or control points, most editing options are available in edit mode for meshes, curves, surfaces and lattices.

Options



Hooks menu

Add, New Empty
: Adds a new hook and create a new empty object, that will be a parent to the selection, at the center of the selection

Add, To Selected Object

When another object is selected (you can do that in edit mode with Ctrl RMB) the new hook is created and parented to that object

Editing Hooks

Mode: Edit Mode

Panel: Editing Context Modifiers

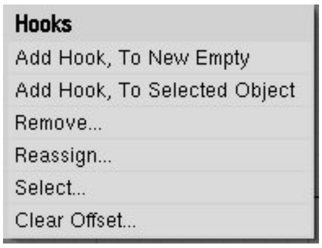
Hotkey: Ctrl H

Menu: Mesh Vertices Add Hook

Description

Once hooks are available in an object, the hook menu will give additional options:

Options



Hooks extended menu

Remove
: This will give a new menu with a list of hooks to remove

Reassign
: Use this if you want to assign new vertices to a hook

Select...
: Select the vertices attached to a specific hook

Clear Offset...
: Neutralize the current transformation of a hook parent

Hook Modifier

Mode: Object Mode / Edit Mode

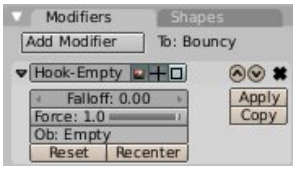
Panel: Editing Context Modifiers

Hotkey: Ctrl H

Description

Hooks are modifiers, that are added to the modifier stack. For each hook modifier, you can give a hook a new name, the default name is the parent name, give it a new parent by typing the new parents name or assign it a Force weighting factor.

Options



In the editing buttons, modifier panel, when a hook is created, you can control it via the panel.

Ob
: The parent object name for the Hook. Changing this name also recalculates and clears offset

Reset
: Recalculate and clear the offset transform of Hook

Recenter
: Set Hook center to cursor position

Select
: Select affected vertices on mesh

Reassign
: Reassigns selected vertices to this hook

Force
: Since multiple hooks can work on the same vertices, you can weight the influence of a hook this way. Weighting rules are:
  - ◊ If the total of all forces is smaller than 1.0, the remainder, 1.0−forces, will be the factor the original position have as force.
  - ◊ If the total of all 'forces' is larger than 1.0, it only uses the hook transformations, averaged by their weights.

Falloff
: If not zero, the falloff is the distance where the influence of a hook goes to zero. It currently uses a smooth interpolation, comparable to the Proportional Editing Tools. (See *mesh_modeling_PET*)

Previous: Manual/Explode Modifier	Contents	*Next:* Manual/Lattice Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

Lattice Modifier

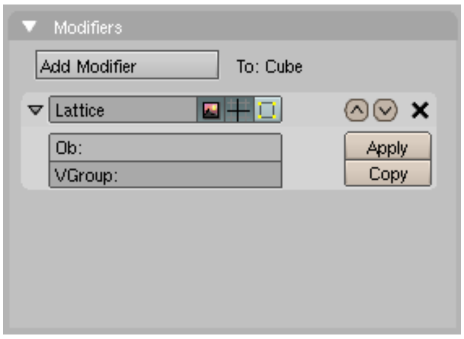
Mode: Object Mode

Panel: Editing Context â†’ Modifiers

Hotkey: F9

The Lattice modifier deforms the base object according to the shape of a Lattice object.

Options



Modifier panel with Lattice modifier activated.

Ob:
: The Lattice object with which to deform the base object.

VGroup:
: An optional vertex group which controls the strength of the deformation.

Hints

The Lattice Modifier has the same functionality as its earlier counterpart. The primary differences between the earlier version and the implementation as a Modifier is that now the effect can be applied in realtime, while editing, to **any** object type without concern for a parent/child relationship.

Instead, one only has to type the name of the lattice object into the Ob: field and the modifier takes effect immediately. You can even see the effect on the object inside edit mode using the Cage Mode (6) button.

There is a separate panel for controlling the Lattice Modifier called Lattice. This panel is where lattice attributes are controlled.

A Lattice consists of a non−renderable three−dimensional grid of vertices. Their main use is to give extra deformation capabilites to the underlying object they control. These child objects can be Meshes, Surfaces and even Particles.

Why would you use a Lattice to deform a mesh instead of deforming the mesh itself in Edit Mode? There are a couple of reasons for that:

- First of all: It's easier. Since your mesh could have a zillion vertices, scaling, grabbing and moving them could be a hard task. Instead, if you use a nice simple lattice your job is simplified to move just a couple of vertices.
- It's nicer. The deformation you get looks a lot better!
- It's fast! You can use the same lattice to deform several meshes. Just give each object a lattice modifier, all pointing to the same lattice.
- It's a good practice. A lattice can be used to get different versions of a mesh with minimal extra work and consumption of resources. This leads to an optimal scene design, minimizing the amount of modelling work. A Lattice does not affect the texture coordinates of a Mesh Surface. Subtle changes to mesh objects are easily facilitated in this way, and do not change the mesh itself.

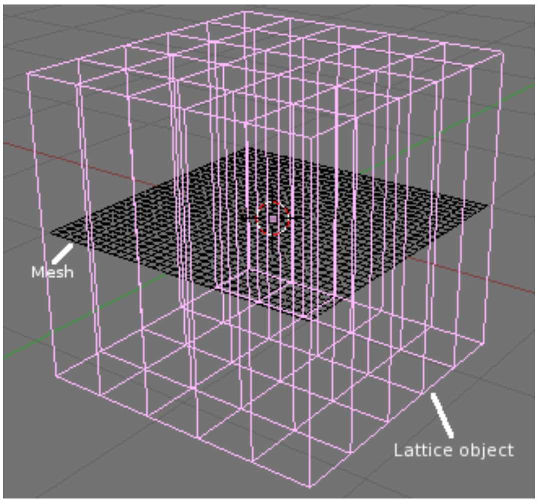
Example/Tutorial(s)

There are example tutorials in the Tutorials section. Ones shows how to shape a fork and the other shows how to make one object follow the shape of another.

Here is a really quick example of creating landscape terrain.

Note

I use the *Top* 3D Window to perform some of these actions.



Lattice and Plane

First start with a Plane Mesh object and subdivide it about 5 times using the "Subdivide" button in the Mesh Tools panel and click "All Edges" so we can see all the faces in wireframe mode.

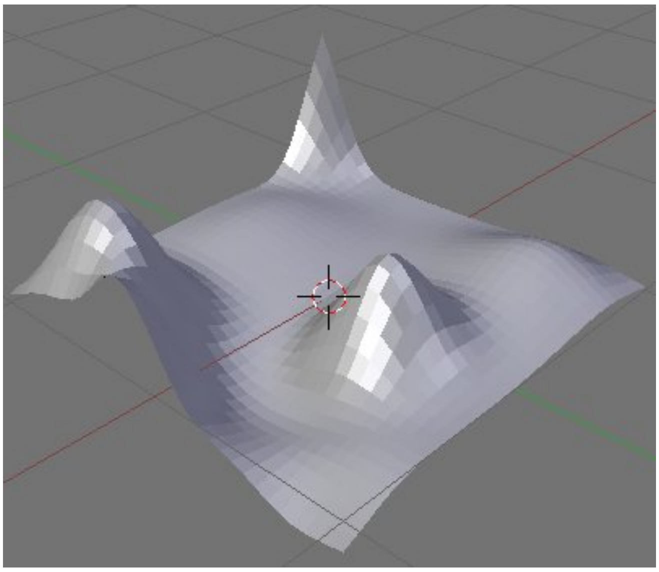
And then add a Lattice object to the scene. The default name for the Lattice object is "Lattice". Scale the Lattice to fit around the Mesh object using the S.

At this time the Lattice and the Mesh are independent of each other. Neither knows of the other. We associate them using the Lattice Modifier. Go ahead and select the Mesh object and *add* a Lattice Modifier to it. Then fill in the "Ob:" field with the name of the Lattice object, which by default is called "Lattice". Be careful the field is case sensitive.

Go back to the Lattice object and bump up the **U** and **V** subdivisions to "**6**" each. (*Lattice and Plane*) is what we have so far as viewed from a perspective 3D window.

Now we can begin to deform the Mesh object by going into Edit Mode while the Lattice object is selected. Grab a vertex, or two, and drag them up the Z−axis (Z). Then grab a few more and move them down the Z−axis. Note, if you feel you don't have enough vertices on the Lattice object you can always exit Edit Mode (i.e. Object mode) and bump up the **U** and **V** subdivisions even more.

(*Quick landscape complete*) is my attempt at creating a landscape. the landscape is in solid mode with the Lattice object hidden on another Layer (M).



Quick landscape complete

Previous: Manual/Hooks	Contents	*Next:* Manual/Mask Modifier

User Manual: Contents | Guidelines | Blender Version **2.48**

Mask Modifier

Mode: ALL

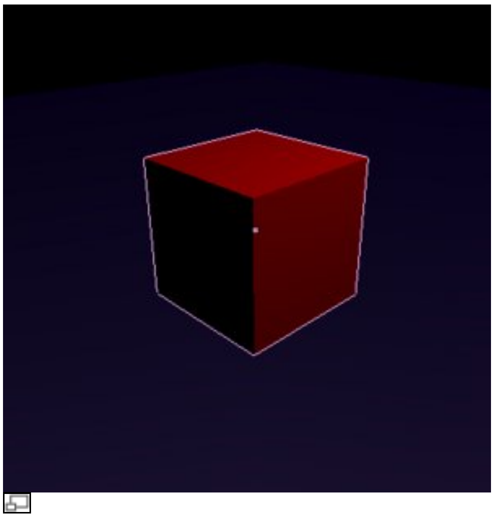
Panel: Editing Modifiers

Hotkey: F9

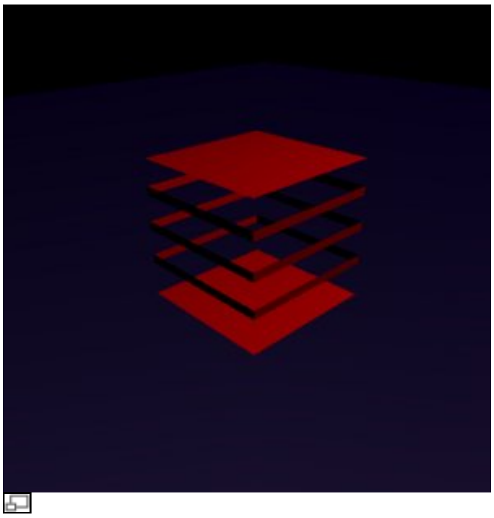
Menu: NONE/NA

Description

The Mask Modifier allows certain parts of an objects mesh to be hidden from view (Masked off), in effect making the parts of the mesh that are masked act as if they were no longer there.



Cube before a Mask Modifier is added to the Cube.

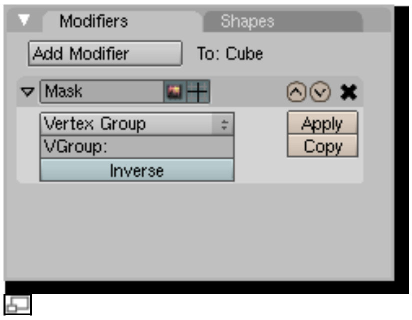


The same Cube with the Mask Modifier added with assigned vertex groups.

The screenshots above show a cube before and after a Mask Modifier is added to it. In the second screenshot the cube has had parts of its mesh assigned to a vertex group which the Mask modifier uses to determine which parts will be visible and which parts wont.

Options

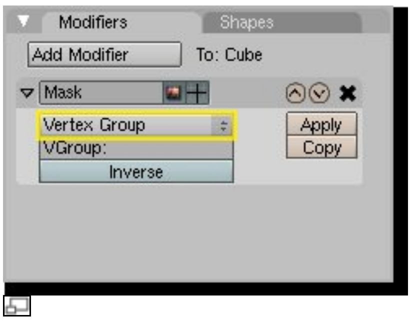
Below is a screenshot of the Mask Modifier panel and some of its options:



Screenshot showing the Mask Modifier panel settings.

- Vertex Group:

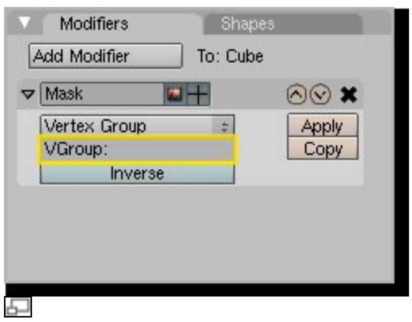
When the Mask Modifier is used it must be told which parts of a mesh are to be affected by the Mask Modifier. There are a few ways to do this, but when the Vertex Group option is selected the Mask Modifier uses a specified Vertex Group to determine which parts of a mesh are Masked by the modifier.



Screenshot showing the Mask Modifier panel settings with the Vertex Group option highlighted in yellow.

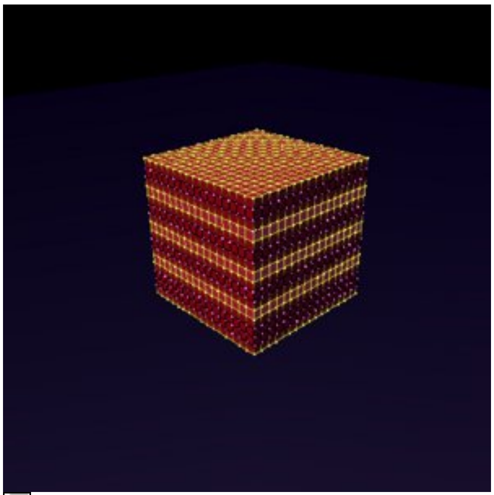
- VGroup:

When the Vertex Group selection option for masking is used the VGroup field becomes visible.

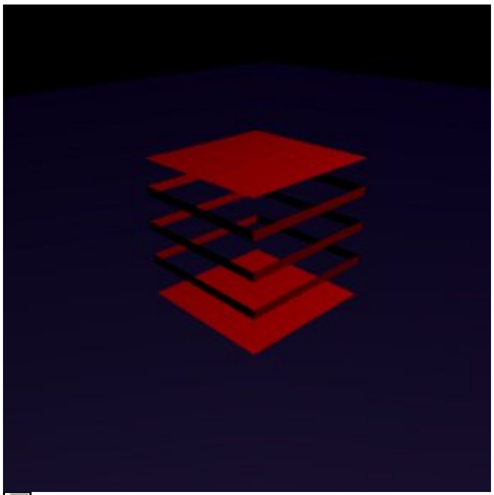


Screenshot showing the Mask Modifier panel settings with the VGroup option highlighted in yellow.

In the VGroup: field enter the name of a vertex group that is associated with the current mesh. When this is done the Mask Modifier will update so that anywhere the vertices of the mesh are part of the named vertex group will be masked (which normally means they will be visible), and anything not part of the named vertex group will be made non−visible.



Cube showing the masked parts of the mesh that are part of a selected vertex group. In this case highlighted in yellow.



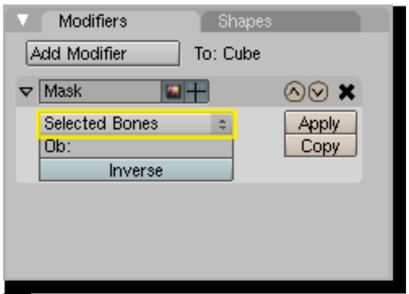
The effect of the vertex group on the underlying Cube with a Mask Modifier active.

The 2 screenshots above show, first the underlying vertex groups assigned to the Cube (highlighted in yellow) and the second image shows the affect on the cube when it has a Mask Modifier applied to it.

Any of the methods for assigning vertex weights to a mesh work with the Mask Modifier, however the actual weight value assigned to a vertex group is ignored by the Mask Modifier. The Mask Modifier only takes into account whether a set of vertices are part of a group or not, the weight is not taken into account. So having a vertex group weight of say .5 will not make a partially masked mesh. Just being part of the vertex group is enough for the Mask Modifier even if the weight is 0.

- Selected Bones:

The Selected Bones information should go here.



Screenshot showing the Mask Modifier panel settings with the Selected Bones option highlighted in yellow.

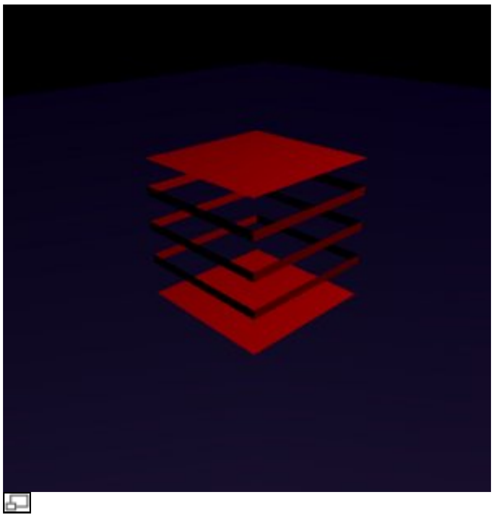
Note

I have been unable to determine how the Bone Selected method for Masking works, so if you know either update this wiki or contact me and I will do it Terrywallwork −− 19 November 2008

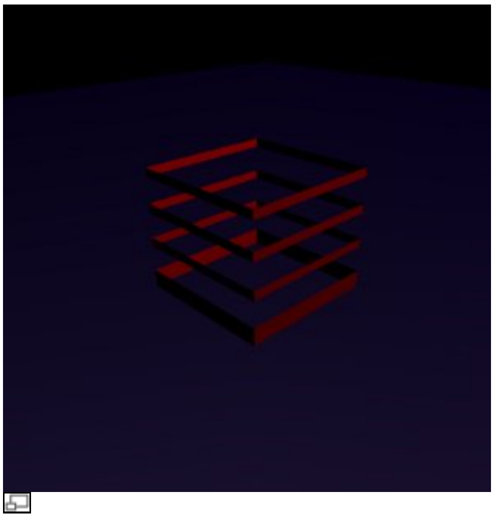
- Inverse:

Normally when the Mask Modifier is applied to areas of a mesh the parts that are under the influence of the Mask Modifier are left visible while the parts that aren't are hidden. The

Inverse button reverses this behavior. In that now parts of the mesh that were not originally visible now become visible and the parts that were visible become hidden.



Screenshot showing the affect on a Cube with a Mask Modifier with the Inverse button not active.



Screenshot showing the affect on a Cube with a Mask Modifier with the Inverse button active.

Examples

Example Blend file using Mask Modifier

See Also

Put related links here.

Previous: Manual/Lattice Modifier	Contents	*Next:* Manual/Mesh Deform Modifier

User Manual: [Contents] | [Guidelines] | Blender Version **2.46**

Mesh Deform Modifier

Mode: ALL

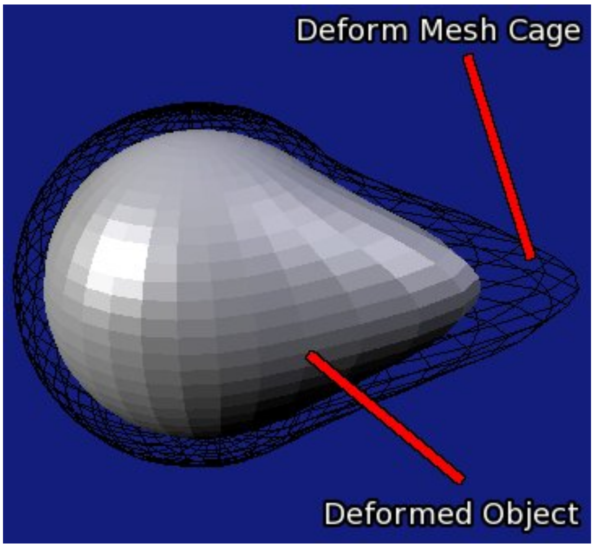
Panel: Editing Modifiers

Hotkey: F9

Menu: NONE/NA

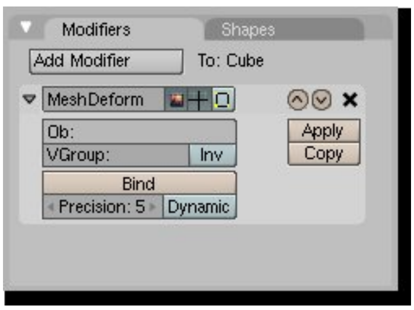
Description

The Mesh Deform Modifier allows an arbitrary closed mesh (of any closed shape, not just a cuboid shape of a Lattice Modifier) to act as a deformation cage around another mesh.



Example of a Deform Mesh Cage − Deforming a Sphere

Above is an example of a Deformed Object (shaded in gray) inside the confines of a Deform Mesh Cage (shown in black wireframe). The Deform Mesh Cage can be any shape of mesh but it must be closed. In the example above the Deform Mesh Cage is just a UVSphere that has been altered using proportional editing, as a result the Deformed Object alters its shape in response. The Deformed Object has had a Mesh Deform Modifier added to it and has been told to use the Deform Mesh Cage as the mesh it will use to deform itself.



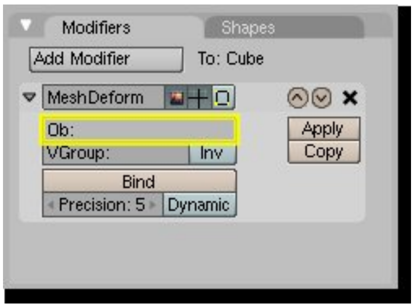
Mesh Deform Modifier Panel

Options

The Mesh Deform Modifier is reasonably easy to use but it can be very slow to do the calculations it needs, to properly map the Deform Mesh Cage to the Deformed Object.

- Ob:

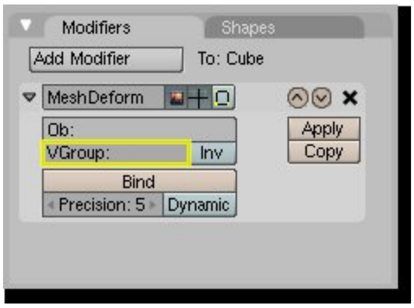
 The Ob: text field is used to indicate which object the Mesh Deform Modifier should use for its Deform Mesh Cage.



Mesh Deform Ob: field highlighted in yellow

- VGroup:

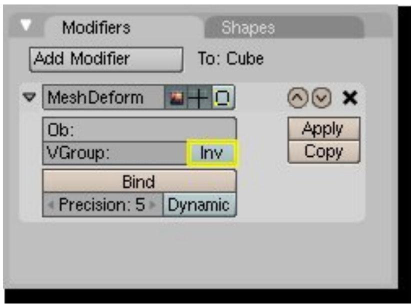
 The VGroup: text field is used to indicate that only the vertices in the specified Vertex Group will be affected by the Deform Mesh Cage.



Mesh Deform VGroup: field highlighted in yellow

- Inv:

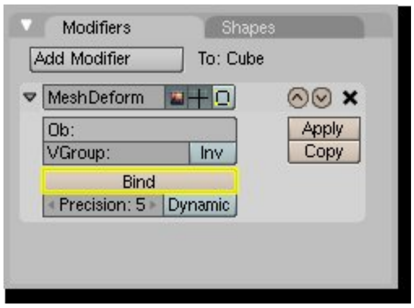
 The Inv toggle button by default is de−activated and means that vertices in the Vertex Group specified in the VGroup: text field will be affected by the Deform Mesh Cage and vertices that aren't in the specified vertex group will NOT be affected. When the Inv toggle button is activated, any vertices within the specified vertex group will NOT be affected by the Deform Mesh Cage, while vertices that aren't in the specified vertex group will be affected by the Deform Mesh Cage. Inv is short for Inverse.



Mesh Deform Inv Button highlighted in yellow

- Bind:

The Bind button is what tells the Mesh Deform Modifier to actually link the Deform Mesh Cage to the Deform Object, so that altering the shape of the Deform Mesh Cage actually alters the shape of the Deform Object.

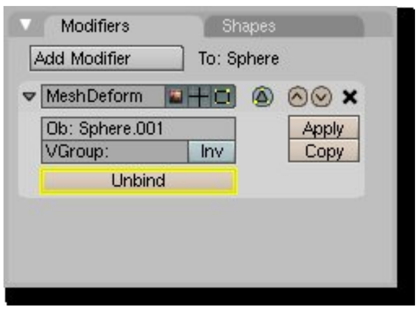


Mesh Deform Bind Button highlighted in yellow

Be aware that depending on the settings of the Mesh Deform Modifier & complexity of the Deform Mesh Cage/Deformed Object, it can take a long time for this operation to complete. This can result in Blender not responding to user actions until it has completed, it is even possible that Blender will run out of memory and crash. As Blender progresses through this operation you should see the top area of the Blender header bar change color as it works its way through Binding the Deform Mesh Cage to the Deformed Object.

- Unbind:

When a Deformed Object has been associated to a Deform Mesh Cage it can later be disassociated with a Deform Mesh Cage by selecting the Unbind button which appears when a Deformed Object has previously been associated with a Deform Mesh Cage.



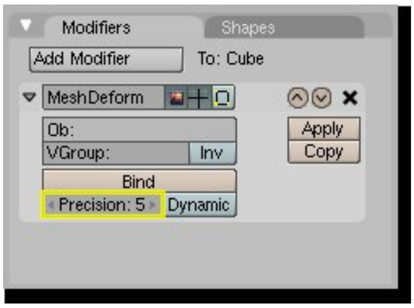
Mesh Deform Unbind Button highlighted in yellow

When Unbind is selected the Deform Mesh Cage will keep its current shape, it will not reset itself back to its original start shape. If the original shape of the Deform Mesh Cage is needed you will have to save a copy of it before you alter it. The Deformed Object will however reset

back to it's original shape that it had before it was Binded to the Deform Mesh Cage.

- Precision:

The Precision numeric slider field, controls that accuracy with which the Deform Mesh Cage alters the Deformed Object when the points on the cage are moved.



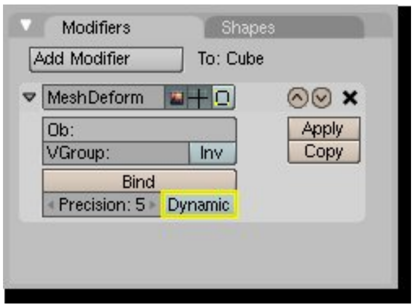
Mesh Deform Precision numeric slider highlighted in yellow

The range of values for the Precision numeric slider field can range from 2 to 10. The default value for the Precision field is 5, raising this value higher can greatly increase the time it takes the Mesh Deform Modifier to complete its Binding calculations.

Increasing Precision slider will get more accurate cage mapping to the Deformed Object. This rise in calculation time can make Blender stop responding until it has calculated what it needs to. As well as making Blender not respond, raising the Precision value high and then trying to Bind on a very complex Deform Mesh Cage or Deformed Object can use large amounts of memory and in extreme cases crash Blender. To be safe, save your blend file before proceeding.

- Dynamic:

The Dynamic button, indicates to the Mesh Deform Modifier that it should also take into account deformations and changes to the underlying Deformed Object which were not a direct result of Deform Mesh Cage alteration.



Mesh Deform Dynamic button highlighted in yellow

With Dynamic button activated other mesh altering features such as other modifiers and Shape Keys are taken into account when Binding a Deform Mesh Cage to a Deformed Object.

The Dynamic button is deactivated by default to save memory and processing time when Binding. When Dynamic is activated deformation quality can be increased, but this comes at the expense of Binding time and memory used.

Other Characteristics of Mesh Deform Modifier

The are some characteristic of the Mesh Deform Modifier which are not directly visible within the Mesh Deform Modifier pane. This section list some of those issues and features.

Mode of Operation

Alterations made to the Deform Mesh Cage will only be reflected in the Deformed Object when the cage is in Edit Mode, when in Object mode the cage can be scaled and distorted but it will not effect the Deformed Object.

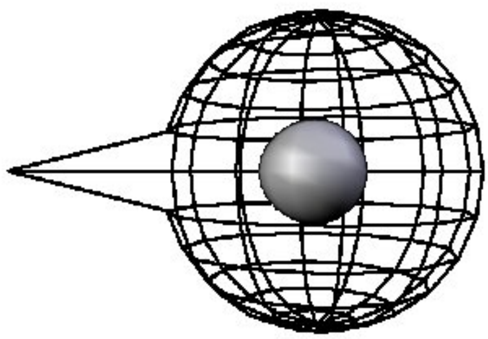
Deform Mesh Cage Location AFTER Binding

While a Deform Mesh Cage is being Binded to a Deformed Object the cage must surround all the parts of the Deformed Object you wish to be affected by the cage. Once the Deform Mesh Cage has been Binded it can be moved away from the Deformed object in Object Mode. When you then switch the Deform Mesh Cage back to Edit Mode and alter its shape, it will alter the Deformed Object even when it is not directly surrounding it.

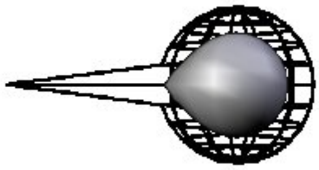
Distance from Deform Mesh Cage & Deformed Object

Distance between the Deform Mesh Cage and the object to be deformed (Deformed Object) has an influence on the amount of change imparted to the Deformed Object when the Deform Mesh Cage is altered (when in Edit Mode). When the Deform Mesh Cage is further away from Deformed Object, then the amount of change imparted to the Deformed Object is less and less local to a specific area of the Deformed Object. When the Deform Mesh Cage is closer to the Deformed Object the amount of influence upon the Deformed Object is greater and more local to a specific area on the Deformed Object.

Un−deformed Sphere



Large Distance Between a Deform Mesh Cage & a Deformed Object



Smaller Distance Between a Deform Mesh Cage & a Deformed Object

Animation showing the difference between each Sphere deform in a Deform Mesh Cage (click to see animation)

Above are examples of the effect of different Deform Mesh Cage distances from a Deformed Object. The top left image shows a normal un−deformed UVSphere. The top right image shows the same UVSphere but with a Deform Mesh Cage which is very close to the Deformed Object, and as a result there is quiet large deformation in the Deformed Object. The bottom left image shows the deformation of a Deformed Object when the Deform Mesh Cage is further away. It can be seen that the Deformed Object alteration is much more muted, even though the vertex that has been moved in the Deform Mesh Cage has moved by the same amount. The bottom left image shows an animated version of the other 3 images showing the change in the Deformed Object with different Deform Mesh Cage distances.

Multi−res support?

At the moment (as of Blender 2.47) Blender does not support use of Multi−res when using the Mesh Deform Modifier.

Interior Control

Besides the cage, having more faces within the cage, which can be loose or forming another smaller cage, can be used for extra control. Such smaller cages may also overlap with the main cage, for example to get extra control over the eyes, two small sphere cages could be added around them.

Interior Control − Clarification?

The above is a paraphrased quote and to me it's not very clear. If anyone has more information or a better way to demonstrate what it is saying, either alter this document or contact me and I will do it. −− Terrywallwork − 25th Sept 2008

Implementation

The Mesh Deform Modifier implementation method in Blender 2.46 is currently "Harmonic Coordinates (for Character Articulation)" Volume Deformation Method developed by Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green and Tom Sanocki of Pixar Animation Studios. This method was presented at the Siggraph 2007 conference. It has many advantages in controlling the deformations of meshes.

A copy of the implementation pdf document can be downloaded here:

Harmonic Coordinates for Character Articulation

A video demonstrating some of the important features of the implementation can be viewed below:

Examples

Blender file showing an example of a Mesh Deform Modifier

Blender file showing the effect of distance on the effect of a Mesh Deform Modifier

Blender file showing the usage of Deform Mesh Cage on the BBB Chinchilla

See Also

Put related links here.

Previous: Manual/Mask Modifier	Contents	*Next:* Manual/Mirror Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

Mirror Modifier

Mode: Any Mode

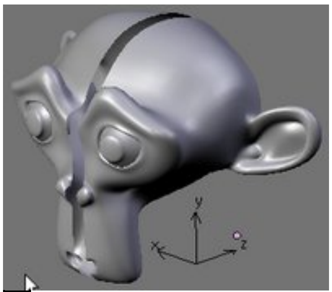
Panel: Editing Context Modifiers

Hotkey: F9

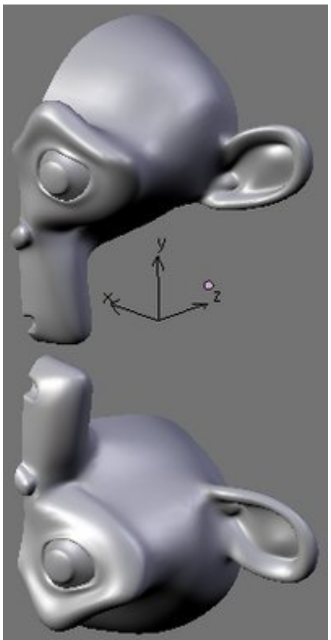
Description

The Mirror Modifier automatically mirrors a mesh along the **Local X**, **Y** or **Z** axis which passes through the object center. It then welds vertices together on the mirror plane within a specified *tolerance* distance. Vertices from the original object can be prevented from moving across or through the mirror plane.

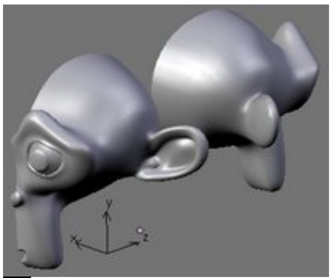
The examples below show the mirror modifier in action in all three axes. The object center, pink dot, is moved from object center so that the mirror plane is more obvious to spot.



Mirror X axis.

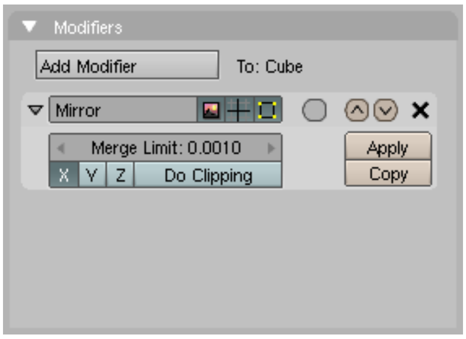


Mirror Y axis.



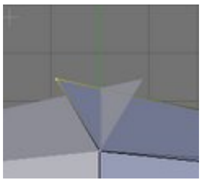
Mirror Z axis.

Options



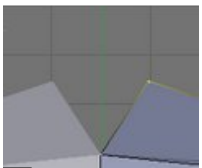
Mirror modifier

- Merge Limit: − The distance before welding of vertices begins. The vertices will snap together when they reach the distance specified.
- X,Y,Z − The axis along which to mirror. To understand how the axis applies to the mirror direction, if you where to mirror on the x axis, the x plus values of the original mesh would become x minus values on the mirrored instance.
- Do Clipping − Prevents vertices from crossing through the mirror plane.

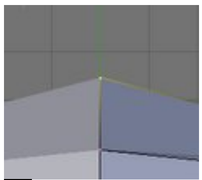


Vertices do not clip together.

While the Do Clipping is un−selected the vertices will not clip together at the mirror plane even if they are within the Merge Limit.



No vertex Clipping.



Vertices have clipped together.

If Do Clipping is selected but vertices are outside of the Merge Limit the vertices will not merge. As soon as the vertices are within Merge Limit they are clipped together and cannot be moved beyond the mirror plane. If several vertices are selected and are at different distances to the mirror plane, they will one by one be clipped at the mirror plane.

Once you have confirmed clipped vertices with LMB you must, if you want to break the clipping, un−select the Do Clipping to be able to move vertices away from the mirror.

Hints

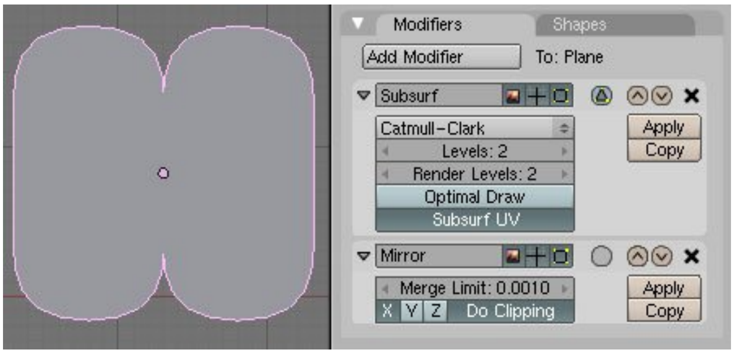
Many modeling tasks involve creating objects that are symmetrical. However there used to be no quick way to model both halves of an object without using one of the workarounds that have been discovered by clever Blender artists over the years.

A common technique is to model one half of an object and use Alt D to create a linked duplicate which can then be mirrored on one axis to produce a perfect mirror−image copy, which updates in realtime as you edit.

The Mirror modifier offers another, simpler way to do this. Once your modeling is completed you can either click Apply to make a real version of your mesh or leave it as is for future editing.

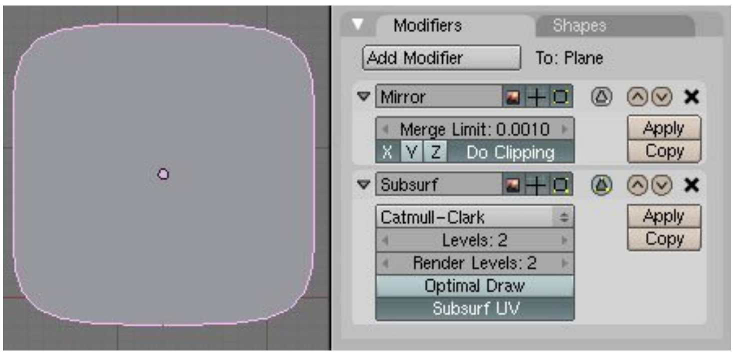
Using Mirror modifier with Subsurf modifier

When using the mirror modifier along with the subsurf mofifier the order in which the modifiers are placed is important.



Subsurf modifier before Mirror modifier

This shows the subsurf modifer placed before the mirror modifier, as you can see the effect of this is that the mesh splits down the centre line of the mirror efect.



Mirror modifier before Subsurf modifier

This shows the mirror modifier placed before the subsurf modifier. In this order you will get the the centre line of the mesh snapped to the centre line, which in most cases would be the desired effect.

Aligning for Mirror

To apply a mirror modifier, it is common to have to move the object's center onto the edge or face that is to be the axis for mirroring. This can be tricky when attempted visually. A good technique to achieve an exact position is to determine the edge against which you wish to mirror. Select two vertices on that edge. Then use Shift S followed by Cursor−>Selection. This will center the 3D cursor exactly on the edge midway between the two vertices. Finally, in the Editing panel (F9), select Center Cursor from the Mesh panel which will move the object's center to where the 3D cursor is located. Now the mirroring will be exact.

Previous: Manual/Mesh Deform Modifier	Contents	*Next:* Manual/Particle Instance Modifier

User Manual: Contents | Guidelines | Blender Version **2.4x**

Particle Instance Modifier

Mode: Any Mode

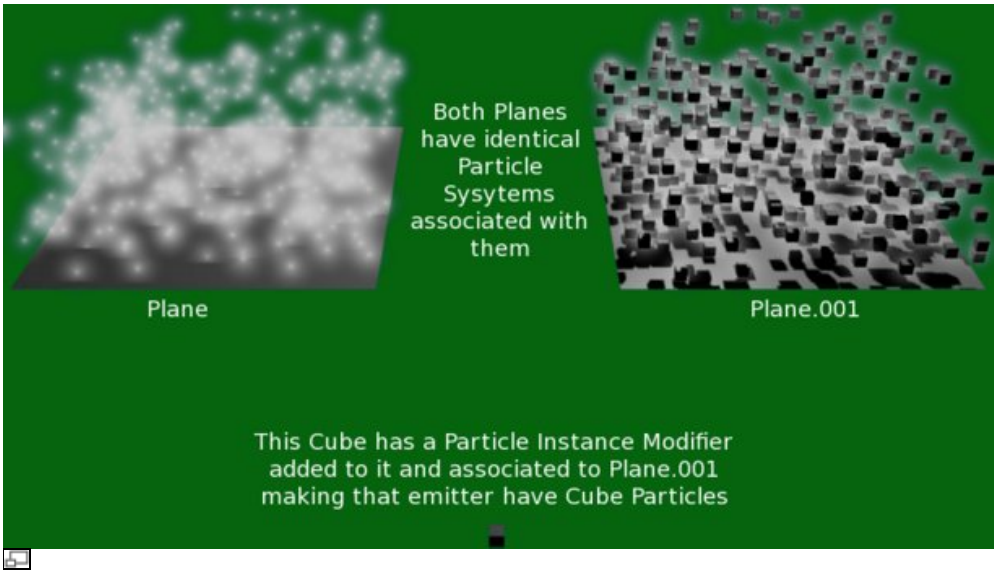
Panel: Editing Context Modifiers

Hotkey: F9

Menu: NOT SET

Description

When a Particle Instance Modifier is added to an object, that object will be used as a particle shape on an object which has a particle system associated with it. This means that to use this Modifier you must also have another object which has a particle system on it, otherwise the Particle Instance Modifier will appear to do nothing.



Particle system on Left has no Particle Instance Modifier object associated with it. The one on the Right is associated with Cube shown by using a Particle Instance Modifier on the Cube

Definitions/Terms

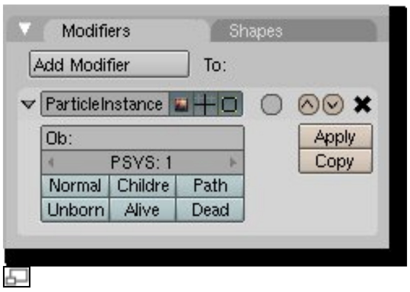
Here is a brief explanation of the various terms and definition used in relation to particles and the Particle Instance Modifier:

- Particle System − Is an object/mesh which has the ability to emit/generate particles activated on it.
- Normal Particle − Is a particle that is not a Children/Child generated particle type.
- Children/Child Particle − Is a particle type that is generated and placed with relation to other Normal particles that already exist. Children/Child particle are generally much quicker to calculate.

- Unborn Particle − Is a particle which has not yet been displayed/emitted because it is not its time to be emitted/displayed. One of the reasons a particle can be in Unborn state is that it is before the frame at which it is to be emitted.
- Alive Particle − Is a particle which has been displayed/emitted and has not yet reached its Dead state. One of the reasons a particle can be in an Alive state is that it has been Alive for less Frames than its life length.
- Dead Particle − Is a particle which has been displayed/emitted and has reached its end of life length and at that point it enters the Dead state.

Options

Because of the co−dependant way in which the Particle Instance Modifier is influenced by the underlying Particle Systems on other objects, some of the apparent effects generated by the Particle Instance Modifier can look and act vastly different, depending on the underlying settings of the Particle Systems it is associated with. This is worth taking account of if the Particle Instance Modifier settings don't appear to be giving the results expected, as it may indicate that the Particle System settings may need altering rather than the Particle Instance Modifier settings.

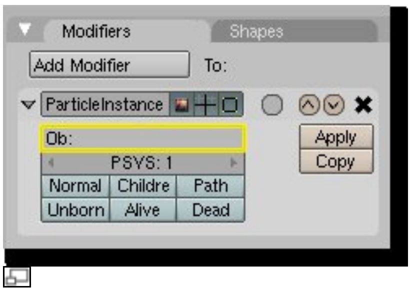


Particle Instance Modifier Panel

- Ob:

  The Ob: field (short of Object), associates this Particle Instance Modifier with another object (usually the other object is the name of an object which has a Particle System associated with it). This indicates that when the object named in this field emits particles, those particles will have the mesh shape of the current Particle Instance Modifier's mesh.

  If for example a sphere has a Particle Instance Modifier added to it, when the Ob: field of that Particle Instance Modifier is filled in with the name of an object that emits particles, those particle will be sphere shaped because the object that had a Particle Instance Modifier added to it was a sphere shaped object.

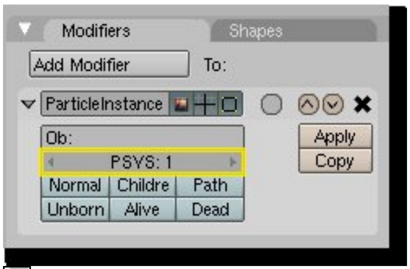


Particle Instance Modifier Panel − With the Ob: field highlighted in yellow.

Even though most of the time the Ob: field will have the name of a object which has a particle system entered into it, it is not mandatory you can enter an object which does not have a particle system and it will be accepted by the Ob: field, as there do not appear to be any checks made to make sure the object name entered into the Ob: field is valid.

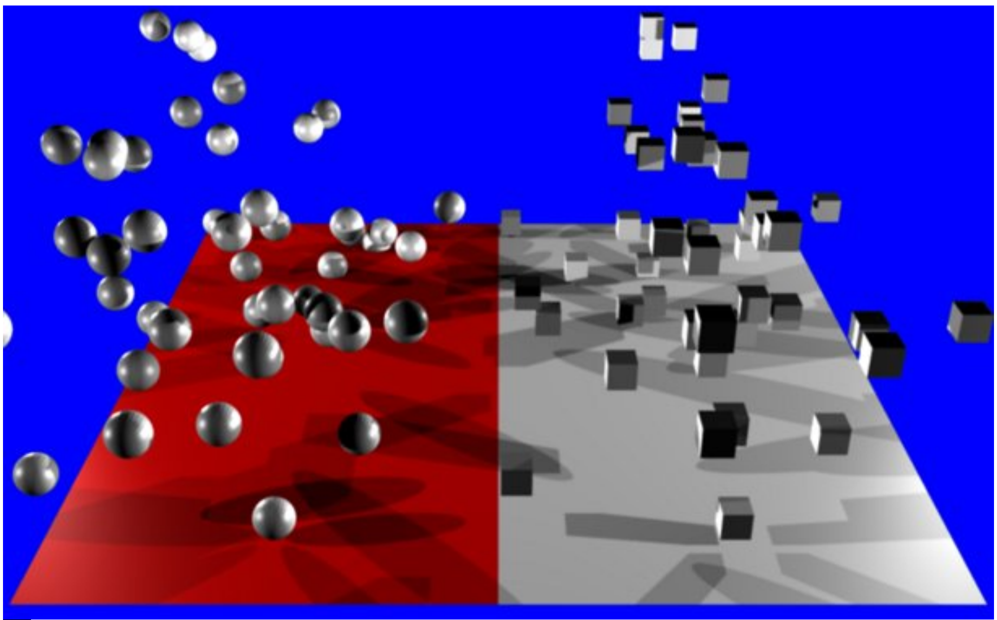
- PSYS:

The PSYS: field (short of Particle System), is used to select which Particle System number to apply the Particle Instance Modifier to, when the mesh which has the Particle System on it has more than one Particle System on it. The PSYS: field can have a value between 1 and 10. It is possible to select any of the 10 Particle System numbers, however a check will NOT be made with the underlying particle emitting object specified previously in the Ob: field. If you select a Particle System number which does not exist on the particle emitting object, then the particles on the emitting mesh will keep their normal particle shapes no warning will be given that the chosen Particle System does not exist on a particular particle emitting mesh.



Particle Instance Modifier Panel − With the PSYS: field highlighted in yellow.

As an example, below is a single plane mesh with 2 areas (the first area shown in Red and the Second in White) with different Particle Systems applied to each area. The left side using a Particle Instance Modifier which has the shape of a sphere and the right side having a Particle Instance Modifier which has the shape of a cube.



Render showing a single Plain mesh object assigned to 2 different vertex groups and each of those vertex groups is assigned a separate and independent Particle System, with each Particle System being assigned a different Particle Instance Modifier. In the case shown the

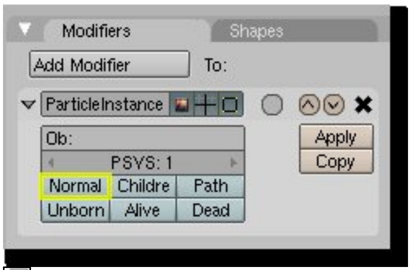
Particle Instance Modifiers are a sphere and a cube.

The Blend file for the example Render above can be obtained here:

Media:Manual − Modifiers − Particle Instance Modifiers − Split Plane.blend

- Normal:

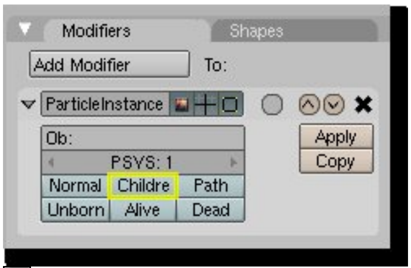
The Normal button (highlighted in yellow) when selected tells the Particle Instance Modifier to draw instances of itself wherever Normal Particle types are emitted from the underlying Particle System, obviously the Particle Instance Modifier has to have been told to have an influence on the underlying particle emitting system (by entering the name of the object to influence in the Particle Instance Modifiers Ob: field). So if the current Particle Instance Modifier is a sphere shape, when Normal particles are emitted a sphere will now appear as well.



Particle Instance Modifier Panel − With the Normal button highlighted in yellow.

- Children:

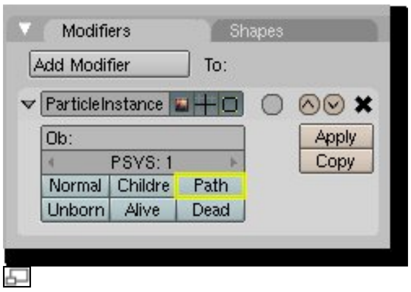
The Children button (highlighted in yellow) when selected tells the Particle Instance Modifier to draw instances of itself wherever Children/Child Particle types are emitted/used on an underlying Particle System, obviously the Particle Instance Modifier has to have been told to have an influence on the underlying particle emitting system (by entering the name of the object to influence in the Particle Instance Modifiers Ob: field). So if the current Particle Instance Modifier is a sphere shape, when Children/Child particles are emitted a sphere will now appear as well.



Particle Instance Modifier Panel − With the Children button highlighted in yellow.

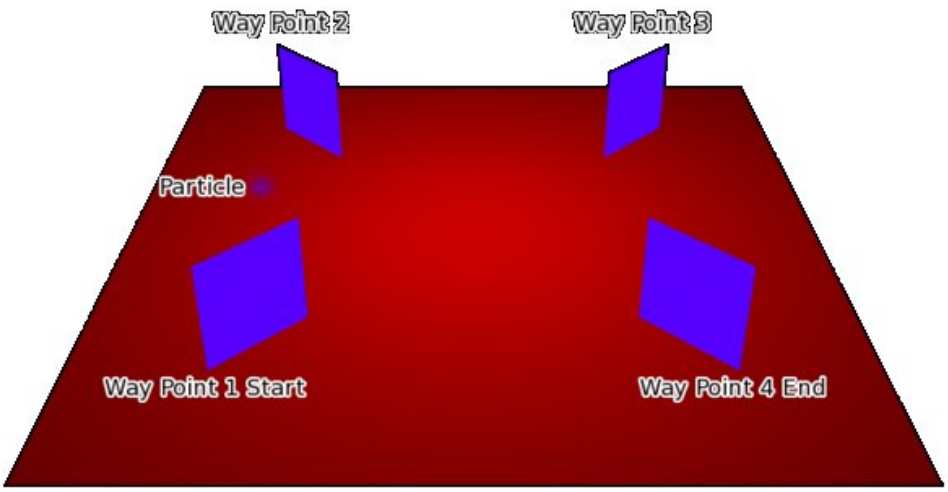
- Path:

The Path button tries to make the mesh object that has a Particle Instance Modifier associated with it, deform its mesh shape in such a way as to try and match the path travelled by the particle/hair strands associated with the mesh object.



Particle Instance Modifier Panel − With the Path button highlighted in yellow.

For example, shown below is a screen shot showing the path of a single Keyed Particle as it travels its way through each of the different way points 1 through 4 (Target Particle Systems), when it reaches way point 4 the particle dies and ends its journey.

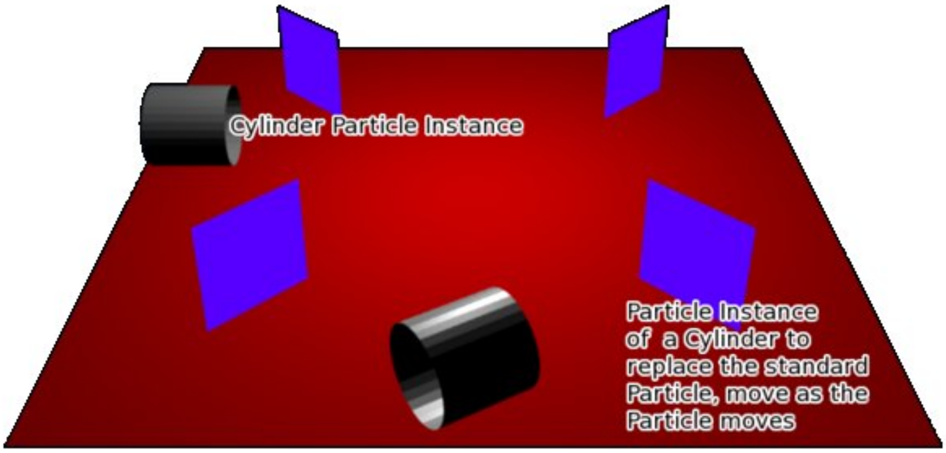


Keyed Particle Following way points showing 1 particle.

The Blend file for the example Render above can be obtained here:

Media:Manual − Particle Instance Modifier − Keyed Particle Example 1.blend

When a Particle Instance Modifier is added to a cylinder object and then associated with the way point 1 particle system, the particle position is copied by the cylinder and placed at the particles position. So the mesh object follows the location of the particle. The cylinder does not alter any of its other properties when following the particle, only the cylinders location gets altered, shape and rotation do not get altered. See screenshot below:



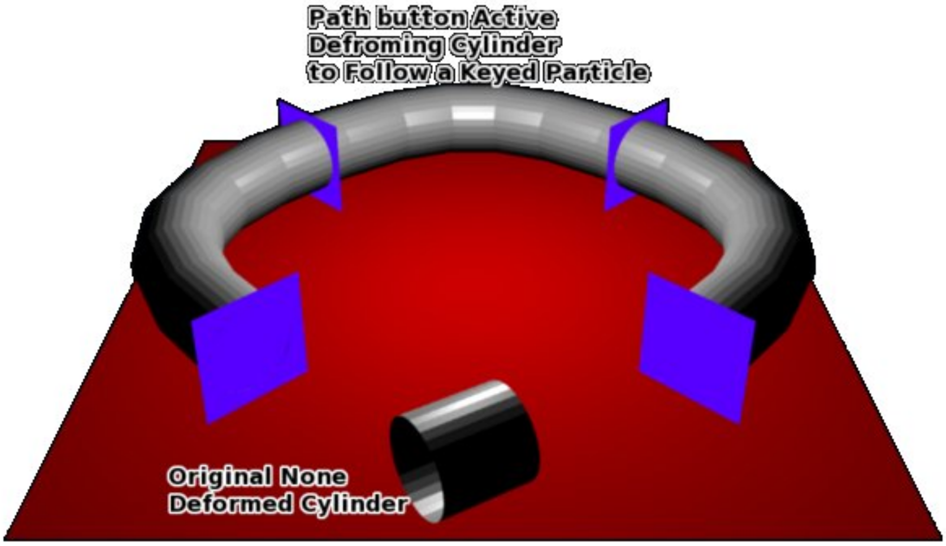
Keyed Particle Following way points showing a mesh object (Particle Instance Modifier) in place of the original particle.

The Blend file for the example Render above can be obtained here:

Media:Manual − Particle Instance Modifier − Keyed Particle Example 2.blend

Both of the above examples had the Particle Instance Modifier Path button deactivated.

When the Path button is activated the effect can be seen in the screenshot below:



Keyed Particle Following way points showing a mesh object (Particle Instance Modifier) in place of the original particle, that is also being deformed to fit the travel path of the original particle.

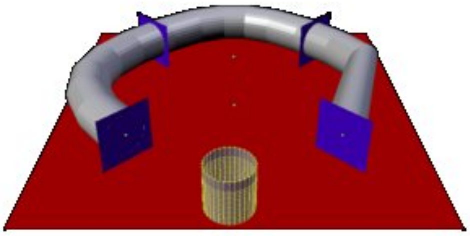
The Blend file for the example Render above can be obtained here:

[Media:Manual − Particle Instance Modifier − Keyed Particle Example 3.blend]

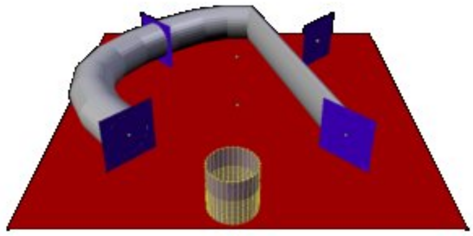
Instead of the cylinder location just following the position of the particle (and not altering its shape), the cylinder tries to fit its mesh to the shape of the path followed by the particle.

The mesh geometry of the object which is trying to deform can have an impact on how well the deformation is carried out. In the case of the cylinder, it has many loop cuts along its length so that it can bend at those points to deform along the particle path.

For example here is the same scene with the number of loop cuts along the length of the cylinder reduced, showing the effect on the deformation of the cylinder along the particle path.

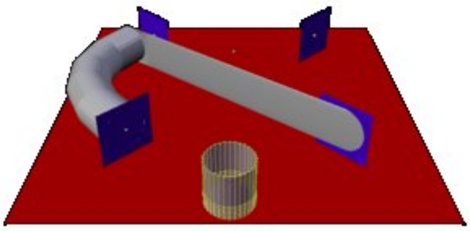


The Cylinder has most of its edge loops so most of the Path



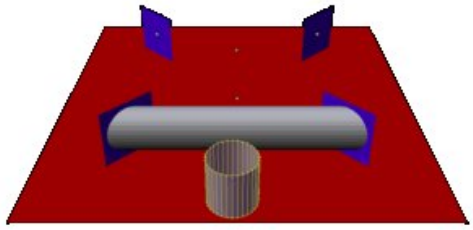
The Cylinder has some of its edge loops

deform is very regular apart from at the very end of the curve.



Now the deform Path is very rough.

removed so the Path of the deform starts to become less regular.



At this point there arn't any vertices to bend the Cylinder to follow the Path and instead the Cylinder just goes directly to the last Way Point 4.

Once all the extra edge loops around cylinder are removed so that there is only the top and bottom vertices left, meaning that the cylinder doesn't have enough geometry to bend, in that case it cannot follow the path of the particle, so it just goes from the start Way Point 1 to the ending Way Point 4.

The Particle Instance Modifier Path button control as well as working for Keyed Particles also work for Hair (strand) Particles. In this case the mesh of the Particle Instance Modifier will follow the length and profile of the hair strands paths.

Below is a screenshot showing the effect of the Path button on hair.

Strand with a Particle Instance Modifier associated with it and deforming the Cylinder along the hair profile.

The Blend file for the example Render above can be obtained here:

Media:Manual − Particle Instance Modifier − Strand Mesh Deform.blend

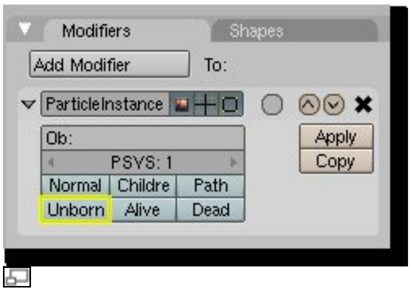
Strands when they are generated instantly die when created so for the Path button to be of any use, you must also have the Dead button selected so that dead strands are displayed, otherwise the path a mesh took will not be visible.

Note

Thanks to Soylentgreen For explaining how the Path button works without his help I would still have been completely lost on how it works. −− Terrywallwork −− 6th Nov 2008.

- Unborn:

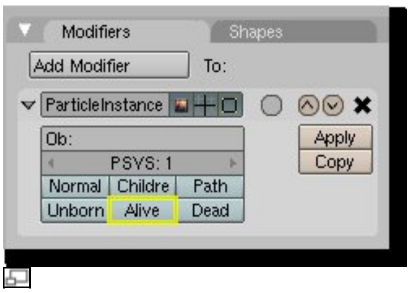
The Unborn button (highlighted in yellow) when selected tells the Particle Instance Modifier to draw instances of itself wherever Unborn Particle types will be emitted/used on an underlying Particle System, obviously the Particle Instance Modifier has to have been told to have an influence on the underlying particle emitting system (by entering the name of the object to influence in the Particle Instance Modifiers Ob: field). So if the current Particle Instance Modifier is a sphere shape, when Unborn particles are present a sphere will now appear as well.



Particle Instance Modifier Panel − With the Unborn button highlighted in yellow.

- Alive:

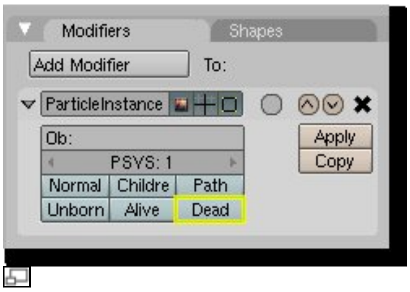
The Alive button (highlighted in yellow) when selected tells the Particle Instance Modifier to draw instances of itself wherever Alive Particle types will be emitted/used on an underlying Particle System, obviously the Particle Instance Modifier has to have been told to have an influence on the underlying particle emitting system (by entering the name of the object to influence in the Particle Instance Modifiers Ob: field). So if the current Particle Instance Modifier is a sphere shape, when Alive particles are present a sphere will now appear as well.



Particle Instance Modifier Panel − With the Alive button highlighted in yellow.

- Dead:

    The Dead button (highlighted in yellow) when selected tells the Particle Instance Modifier to draw instances of itself wherever Dead Particle types will occur on an underlying Particle System, obviously the Particle Instance Modifier has to have been told to have an influence on the underlying particle emitting system (by entering the name of the object to influence in the Particle Instance Modifiers Ob: field). So if the current Particle Instance Modifier is a sphere shape, when Dead particles are present a sphere will now appear as well.



Particle Instance Modifier Panel − With the Dead button highlighted in yellow.

Examples

Examples of Usage and Tutorials here.

See Also

Related topics here.

Previous: Manual/Mirror Modifier	Contents	*Next:* Manual/Smooth Modifier

User Manual: Contents | Guidelines | Blender Version **2.44**

Smooth Modifier

Mode: Any Mode

Panel: Editing Context Modifiers

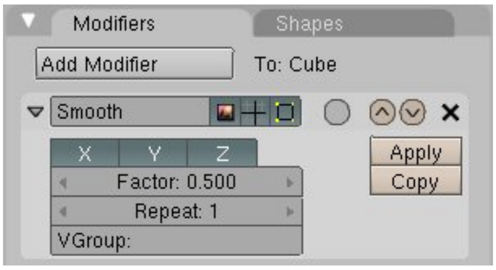
Hotkey: F9

Description

This modifier smooths a mesh by flattening the angles between adjacent faces in it, just like the Smooth button in the Editing Context Mesh Tools panel. So it smooths without subdividing the mesh −− the number of vertices stays the same.

This modifier is not limited to smoothing, though. Its control factor can be configured outside the [0.0, 1.0] range (including negative values), which can result in interesting deformations, depending on the affected mesh.

Options



Smooth modifier

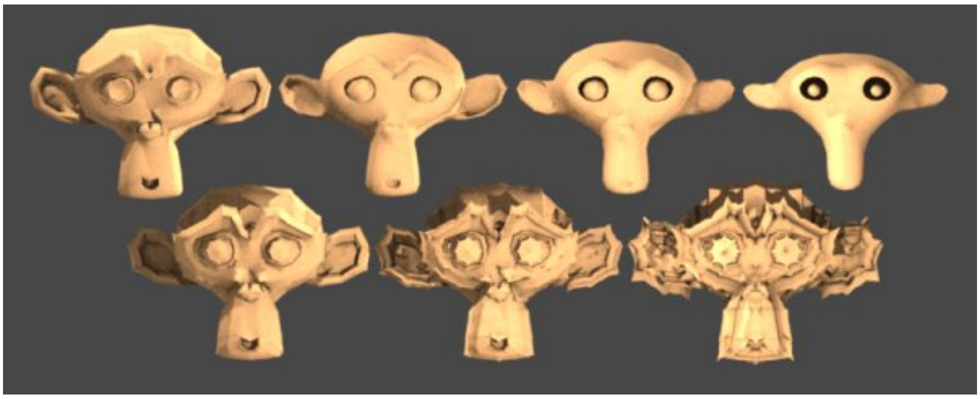
X Y Z
: Toggle buttons to enable / disable the modifier in the X, Y, Z axes directions.

Factor
: The factor to control smoothing amount: The smoothing range is [0.0, 1.0]: 0.0: disable, 0.5: same as the Smooth button, 1.0: max. Alternatively, values outside this range (above 1.0 or below 0.0) distort the mesh.

Repeat
: The number of smoothing iterations, equivalent to pressing the Smooth button (link) multiple times.

VGroup
: A vertex group name, to restrict the effect to the vertices in it only. This allows for selective, realtime smoothing, by painting vertex weights.

Examples



Smooth modifier examples. Option "repeat" is set as 5 for all models, but each one has a different "factor" value. From left to right: **Top** (smoothing): 1) normal Suzanne (same as factor: 0.0); 2) Factor: 0.5; 3) 1.0; 4) 2.0. **Bottom** (distortion): 1) Factor: −0.3; 2) −0.6; 3) −1.0. The three models at the bottom row were subdivided twice to look nicer.

Previous: Manual/Particle Instance Modifier	Contents	*Next:* Manual/Softbody Modifier

Softbody Modifier

If a mesh has been designated as a Softbody, then the simulation engine will be changing the shape of the mesh. When the mesh is designated as a Soft Body, a modifier is inserted in its modifier stack. The location of this modifier in the stack is important, since the Softbody shape can be modified before or after by another modifier, such as an armature. If the Armature modifier is above the Softbody modifier, the Armature will deform the mesh first, and then the Softbody will simulate gravity and mass motion. If reversed, the whole mesh will drop first under the Softbody simulation, and then will be moved by the Armature (generally not desired).

Previous: Manual/Smooth Modifier	Contents	*Next:* Manual/Subsurf Modifier

User Manual: [Contents] | [Guidelines] | Blender Version **2.45**

Subdivision Surfaces

Mode: Any Mode

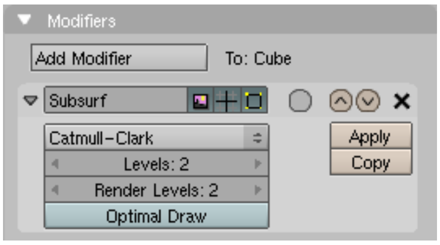
Panel: Editing Context Modifiers

Hotkey: F9 (Panel) Shift O (Toggle SubSurf in Object Mode)

Description

A Subdivision Surface is a method of subdividing the faces of a mesh to give a smooth appearance, to enable modelling of complex smooth surfaces with simple, low−vertex meshes. This allows high resolution Mesh modelling without the need to save and maintain huge amounts of data and gives a smooth *organic* look to the object. With any regular Mesh as a starting point, Blender can calculate a smooth subdivision on the fly, while modelling or while rendering, using Catmull−Clark Subdivision Surfaces or, in short SubSurf.

Options



Modifiers panel.

SubSurf is a Modifier. To add it to a mesh, press Add Modifier and select Subsurf from the list.

- Levels defines the display resolution, or level of subdivision.
- Render Levels is the levels used during rendering. This allows you to keep a fast and lightweight approximation of your model when interacting with it in 3D, but use a higher quality version when rendering.
- To view and edit the results of the subdivision ("isolines") on the Editing Cage while you're editing the mesh, click in the gray circle next to the arrows for moving the modifier up and down the stack. This lets you grab the points as they lie in their new subdivided locations, rather than on the original mesh.
- Optimal Draw restricts the wireframe display to only show the original mesh cage edges, rather than the subdivided result to help visualisation.

Hints

You can use Shift O if you are in ObjectMode to switch Subsurf On or Off. To turn the subsurf view off (to reduce lag), press Alt+Shift+O. The Subsurf level can also be controlled via Ctrl 1 to Ctrl 4, but this only affects the visualization sub−division level.

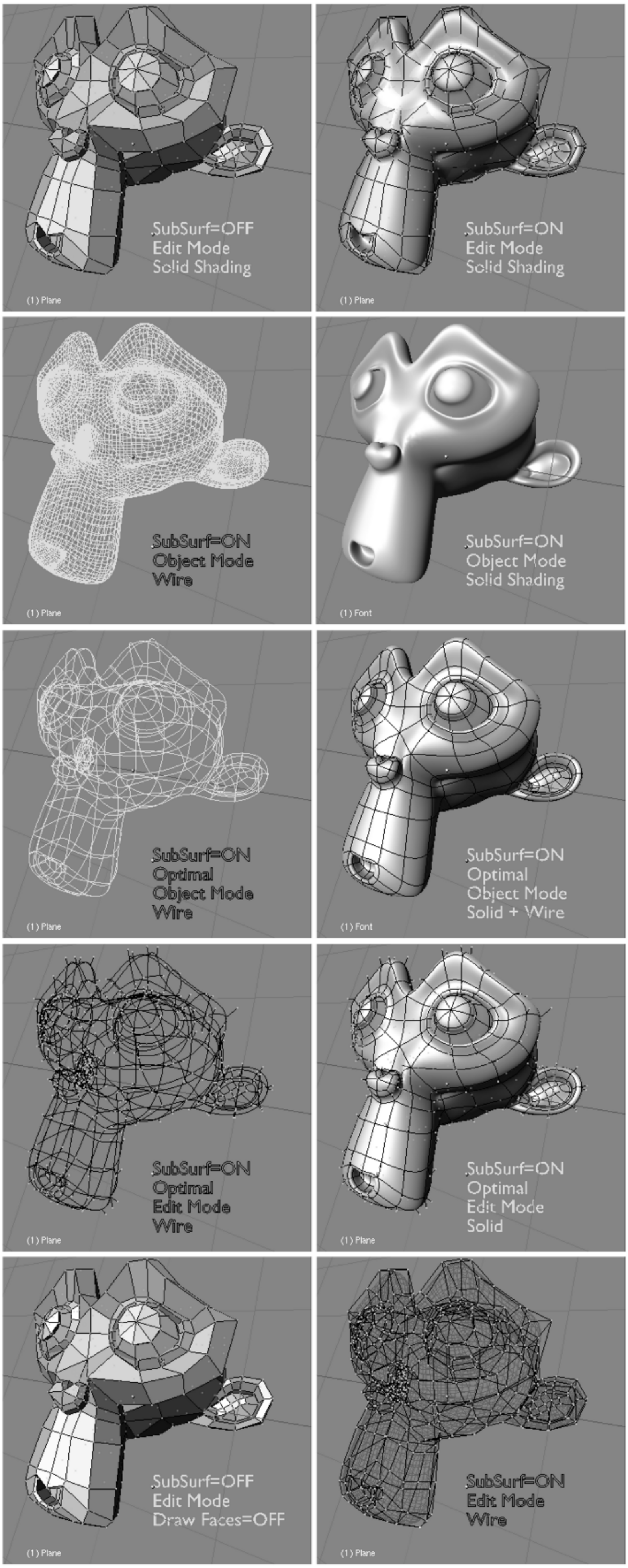
A SubSurfed Mesh and a NURBS surface have many points in common such as they both rely on a "coarse" low−poly "mesh" to define a smooth "high definition" surface, however, there are notable differences:

- NURBS allow for finer control on the surface, since you can set "weights" independently on each control point of the control mesh. On a SubSurfed mesh you cannot act on weights.
- SubSurfs have a more flexible modelling approach. Since a SubSurf is a mathematical operation occurring on a mesh, you can use all the modelling techniques described in this chapter on the mesh. There are more techniques, which are far more flexible, than those available for NURBS control polygons.

Since Subsurf computations are performed both real−time, while you model, and at render time, and they are CPU intensive, it is usually good practice to keep the SubSurf level low (but non−zero) while modelling; higher while rendering.

Examples

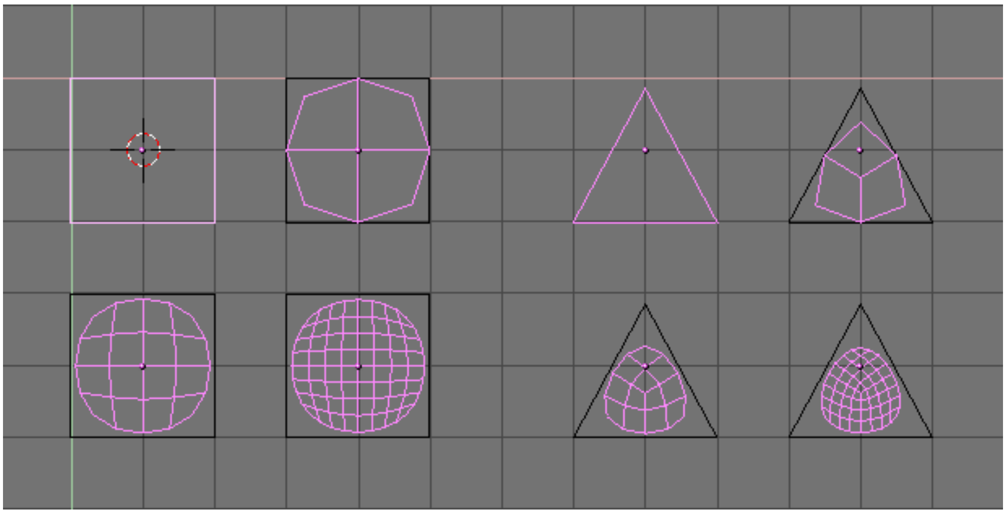
SubSurfed Suzanne shows a series of pictures showing various different combinations of Subsurf options on a Suzanne Mesh.


SubSurf=OFF
Edit Mode
Solid Shading
(1) Plane
SubSurf=ON
Edit Mode
Solid Shading
(1) Plane
SubSurf=ON
Object Mode
Wire
(1) Plane
SubSurf=ON
Object Mode
Solid Shading
(1) Font
SubSurf=ON
Optimal
Object Mode
Wire
(1) Plane
SubSurf=ON
Optimal
Object Mode
Solid + Wire
(1) Font
SubSurf=ON
Optimal
Edit Mode
Wire
(1) Plane
SubSurf=ON
Optimal
Edit Mode
Solid
(1) Plane
SubSurf=OFF
Edit Mode
Draw Faces=OFF
(1) Plane
SubSurf=ON
Edit Mode
Wire
(1) Plane

SubSurfed Suzanne.

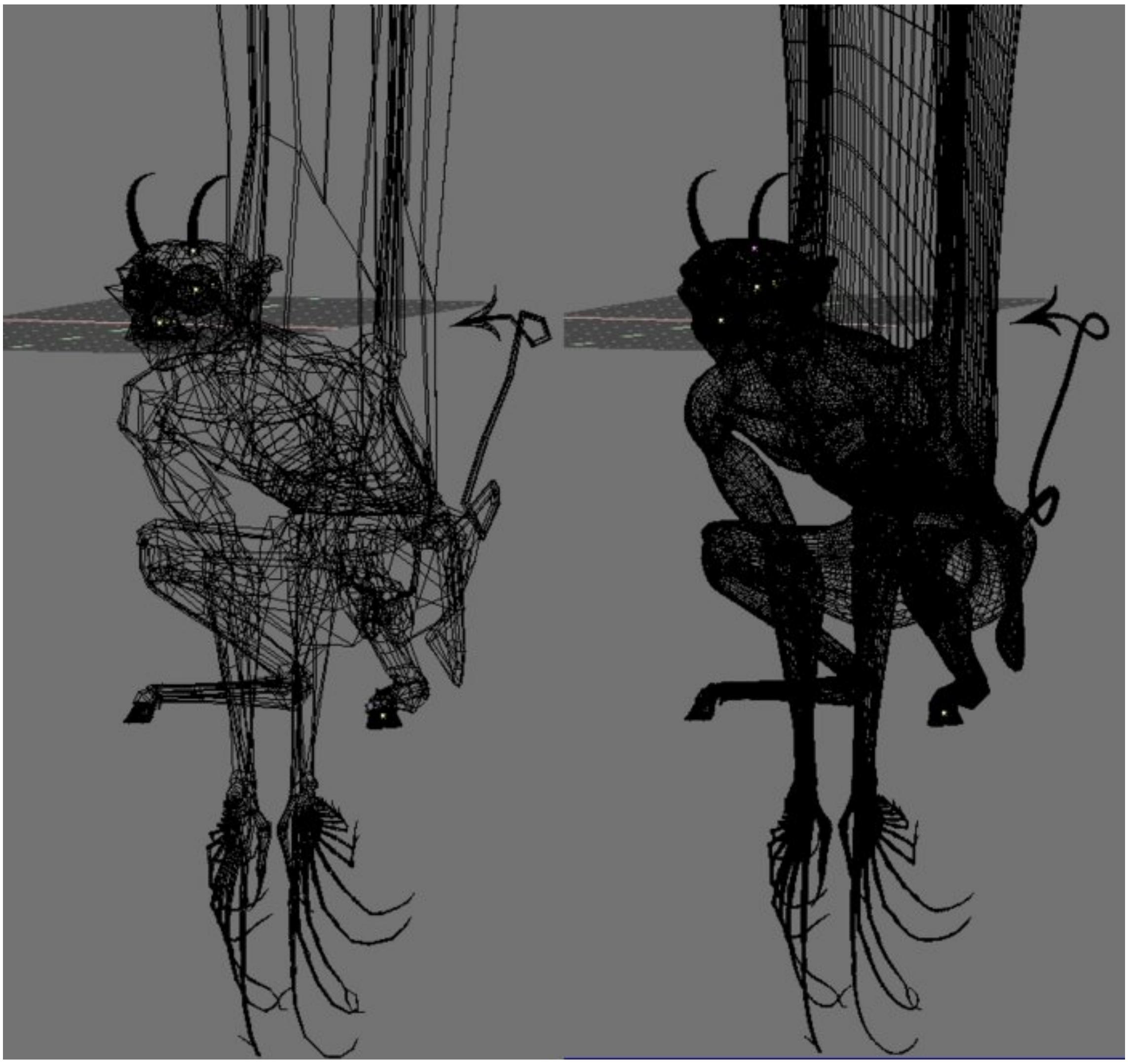
SubSurf of simple square and triangular faces. shows a 0,1,2,3 level of SubSurf on a single square face or on a single triangular face. Such a subdivision is performed, on a generic mesh, for *each* square or triangular face.

It is evident how each single quadrilateral face produces 4^n faces in the SubSurfed mesh. n is the SubSurf level, or resolution, while each triangular face produces $34^{(n-1)}$ new faces (*SubSurf of simple square and triangular faces.*). This dramatic increase of face (and vertex) number results in a slow−down of all editing, and rendering, actions and calls for lower SubSurf level in the editing process than in the rendering one.

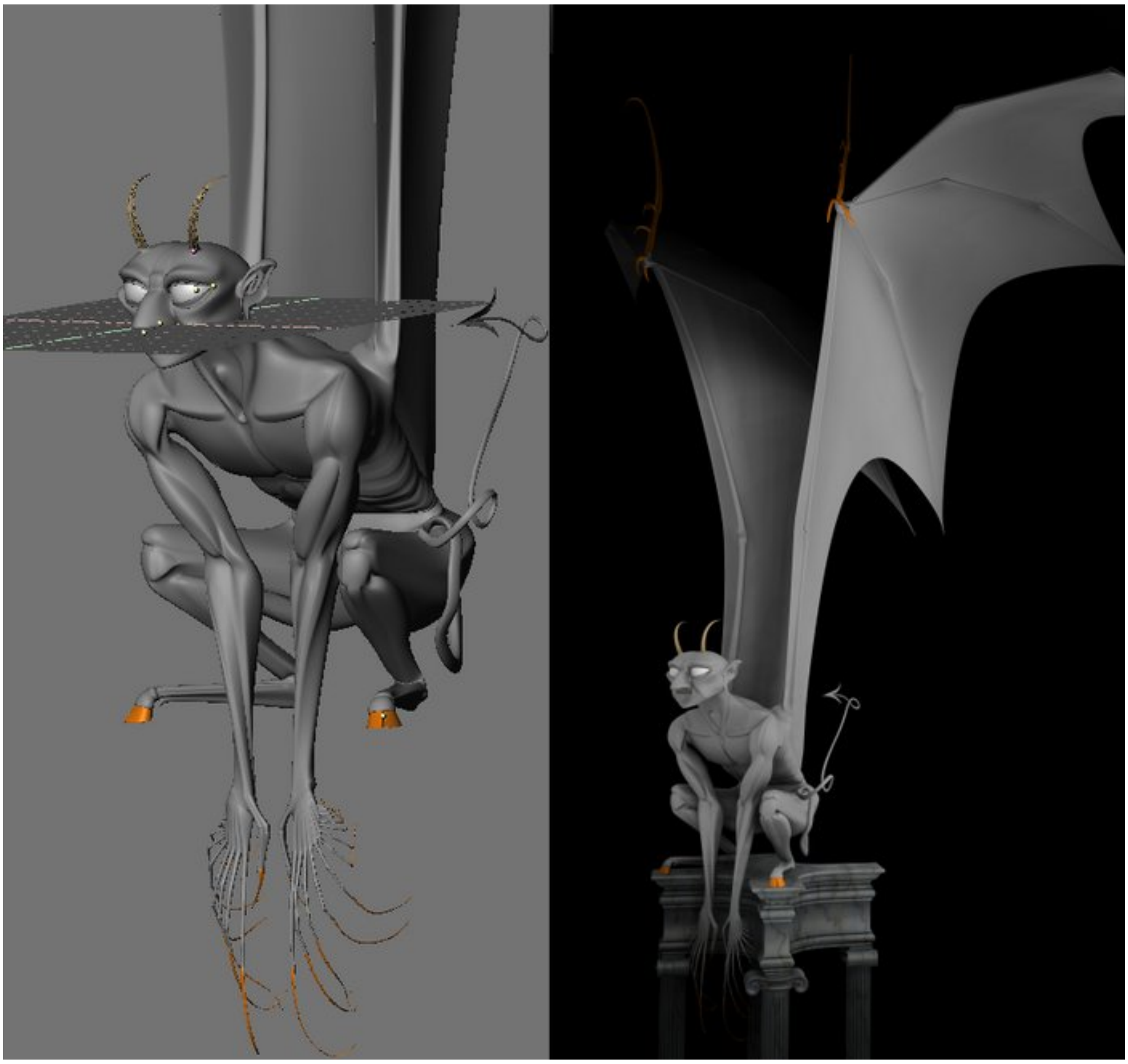


SubSurf of simple square and triangular faces.

The SubSurf tool allows you to create very good "organic" models, but remember that a regular Mesh with square faces, rather than triangular ones, gives the best results. *A Gargoyle base mesh (left) and pertinent level 2 SubSurfed Mesh (right).* and *Solid view (left) and final rendering (right) of the Gargoyle.* show an example of what can be done with Blender SubSurfs.



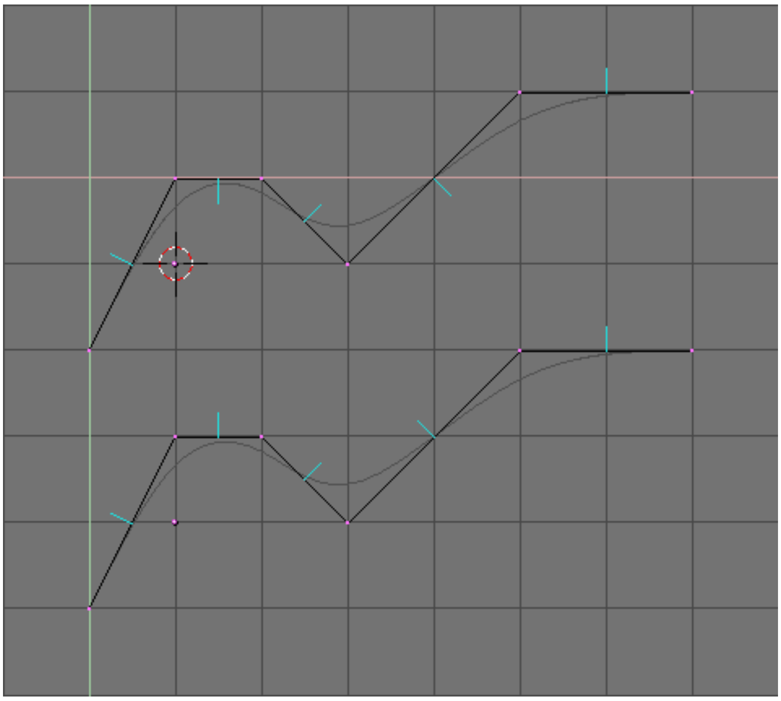
A Gargoyle base mesh (left) and pertinent level 2 SubSurfed Mesh (right).



Solid view (left) and final rendering (right) of the Gargoyle.

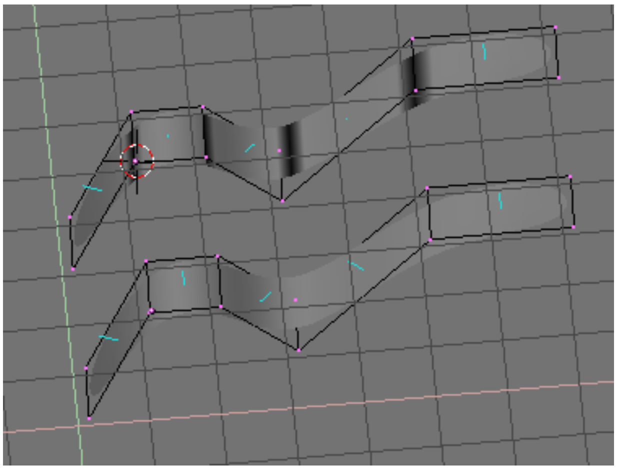
Limitations & Work−arounds

Blender's subdivision system is based on the Catmull−Clark algorithm. This produces nice smooth SubSurf meshes but any 'SubSurfed' face, that is, any small face created by the algorithm from a single face of the original mesh, shares the normal orientation of that original face.



Side view of subsurfed meshes. With random normals (top) and with coherent normals (bottom)

This is not an issue for the shape itself, as *Side view of subsurfed meshes. With random normals (top) and with coherent normals (bottom)* shows, but it is an issue in the rendering phase and in solid mode, where abrupt normal changes can produce ugly black lines (*Solid view of SubSurfed meshes with inconsistent normals (top) and consistent normals (bottom).*).

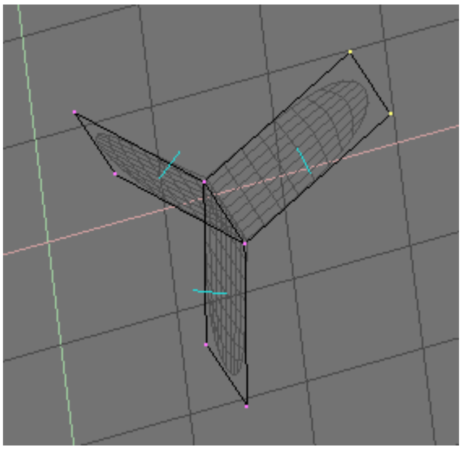


Solid view of SubSurfed meshes with inconsistent normals (top) and consistent normals (bottom).

Use the Ctrl N command in EditMode, with all vertices selected, to recalculate the normals to point outside.

In these images the face normals are drawn cyan. You can enable drawing normals in the EditButtons (F9) menu.

Note that Blender cannot recalculate normals correcty if the mesh is not "Manifold". A "Non−Manifold" mesh is a mesh for which an 'out' cannot unequivocally be computed. From the Blender point of view, it is a mesh where there are edges belonging to *more* than two faces.



A "Non−Manifold" mesh

A "Non−Manifold" mesh shows a very simple example of a "Non−Manifold" mesh. In general a "Non−Manifold" mesh occurs when you have internal faces and the like.

A "Non−Manifold" mesh is not a problem for conventional meshes, but can give rise to ugly artifacts in SubSurfed meshes. Also, it does not allow decimation, so it is better to avoid them as much as possible.

Use these two hints to tell whether a mesh is "Non Manifold":

- The Recalculation of normals leaves black lines somewhere
- The "Decimator" tool in the Mesh Panel refuses to work stating that the mesh is "Non Manifold"

Weighted creases for subdivision surfaces

Mode: Edit Mode (Mesh)

Panel: 3D View Transform Properties

Hotkey: Shift E or N (Transform Properties Panel)

Menu: Mesh Edges Crease Subsurf

Description

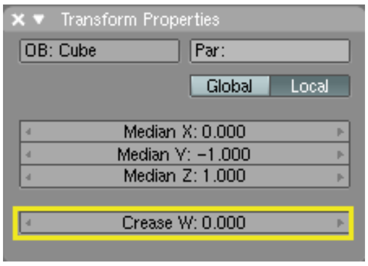
Weighted edge creases for subdivision surfaces allows you to change the way Subsurf subdivides the geometry to give the edges a smooth or sharp appearance.

Options

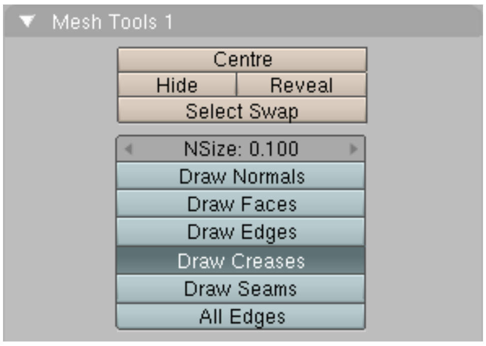
The crease weight of selected edges can be changed interactively by using Shift E and moving the mouse towards or away from the selection. Moving the mouse away from the edge increases the weight. You can also use Transform Properties (N) and enter the value directly. A higher value makes the edge "stronger" and more resistant to subsurf. Another way to remember it is that the weight refers to the edge's sharpness. Edges with a higher weight will be deformed less by subsurf. Recall that the subsurfed shape is a product of all

intersecting edges, so to make the edges of an area sharper, you have to increase the weight of all the surrounding edges.

You can enable an indication of your edge sharpness by enabling Draw Creases. See (*Mesh Tools 1 panel*).



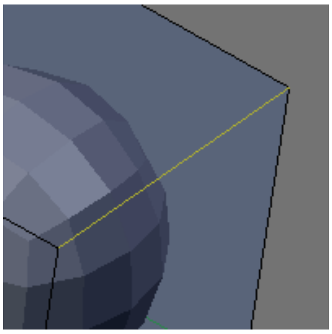
Transform Properties

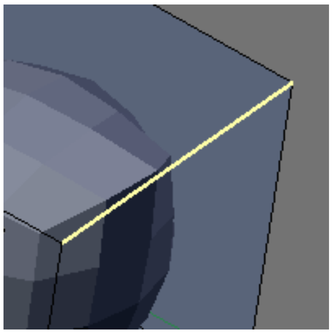


Mesh Tools 1 panel

Examples

The sharpness value on the edge is indicated as a variation of the brightness on the edge. If the edge has a sharpness value of 1.0, the edge will have a brighter color, and if sharpness value is 0.0, the edge will not be so bright.


Edge sharpness 0.0


Edge sharpness 1.0

Previous: Manual/Softbody Modifier | Contents | *Next:* Manual/UVProject Modifier

If you arrived at this page from either pages "Weight Paint" or "Multi Resolution Mesh" and wish to carry on navigating in the order laid out in the Mesh section of the Main Index page use the Navigation bar below:

Previous: Manual/Weight Paint | Contents | *Next:* Manual/Multires

User Manual: Contents | Guidelines | Blender Version **2.4x**

UVProject Modifier

Mode: Any Mode

Panel: Editing Context Modifiers

Hotkey: F9

Description

Ok, so have you ever seen someone walk in front of the slide projector, and the slide is projected on them as they walk in front of it? Well, this modifier does that to the object, using an image (still, movie clip, etc) that you specify.

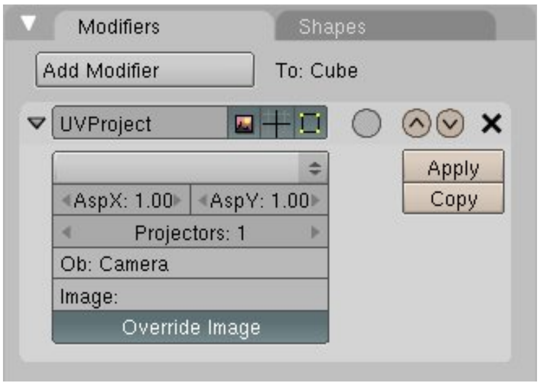
The UVProject modifier projects a UV−mapped image onto the base mesh as if from a slide projector. It does this by altering the base mesh's UV coordinates as though they were projected onto the object by one or more projector objects. The projection used is either an orthogonal or perspective projection along the projector object's negative Z axis.

The effect can be limited to just those faces associated with a given image, or all faces can be affected. The modifier can also override the image assigned to each face with a given image.

For the results to be visible in the final render, the base mesh must have a

- material that uses the affected UV coordinate layer in some way (e.g. by having TexFace enabled).
- the object must have the named UV Texture active

Options



UVProject modifier

UV Layer
: Selects the UV coordinate layer to affect. If no UV layers are present, the modifier will be disabled.

AspX, AspY
: Sets the aspect ratio of the image being projected.

Projectors

The number of projector objects to use (from 1 to 10). An Ob field will be added to the panel for each projector.

Ob

A projector object to use. There will be from 1 to 10 of these fields, depending on the value of Projectors. For each face, the projector whose projection axis is most perpendicular to that face will be used.

- ◊ If a projector object is a camera, the projection used will depend on the camera type: orthographic projection for orthographic cameras and perspective projection for perspective cameras.
- ◊ If a projector object is not a camera, the projection will always be orthographic.

Image

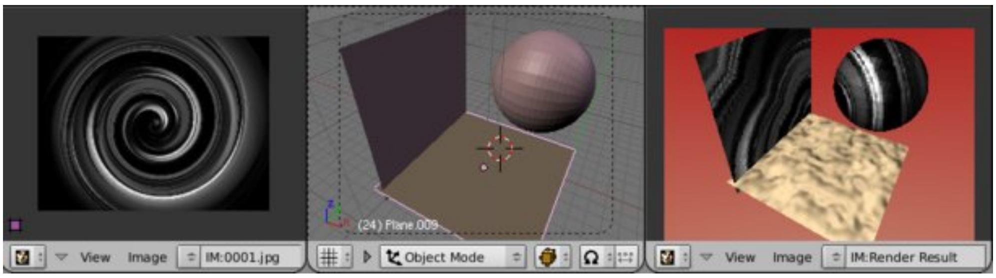
The image to use.

- ◊ If Image is empty, **all faces** are affected
- ◊ If Image is not empty, **only faces with that image assigned** are affected (unless Override Image is on; see below)

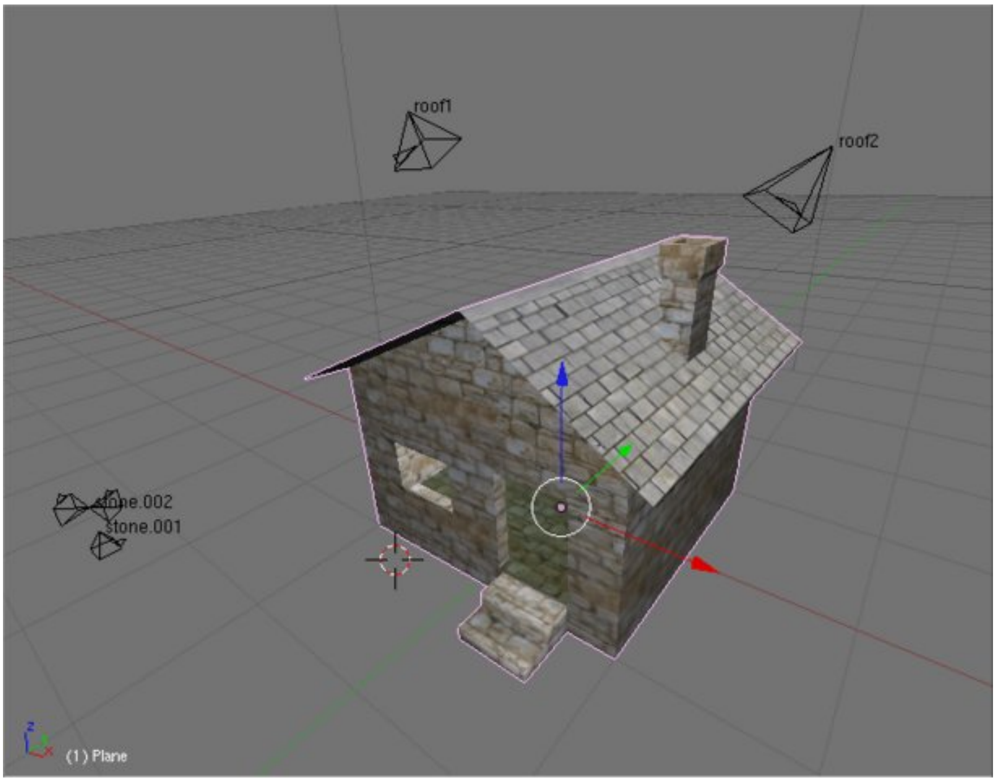
Override Image

This button controls whether to override the current face image with the contents of Image. If Override Image is on, **all faces** will be affected by the modifier, and will have the contents of Image assigned to them (if Image is blank, faces will have no image assigned).

Examples



A swirl image is loaded in the UV/Image Editor and thus known to Blender. A simple scene, shown in the middle, has a wall and a sphere with the modifier set to project the swirl from the 35mm (perspective) camera onto those two objects. The two objects have the same material as the floor, but their colors are superseded by the modifier. As the sphere moves (not shown), the part of the image that it takes moves correspondingly. Kinda spooky, if you ask me.



A house textured with the UVProject modifier. The cameras scattered around the scene are the projection objects. UVProject greatly simplifies this kind of texturing.
Sample blend file

Previous: Manual/Subsurf Modifier	Contents	*Next:* Manual/Wave Modifier

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.4x**

Wave Modifier

Mode: Object Mode

Panel: Editing Context Modifiers

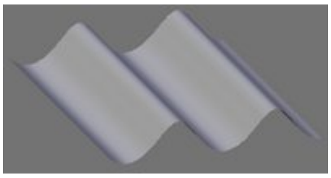
Hotkey: F9

Description

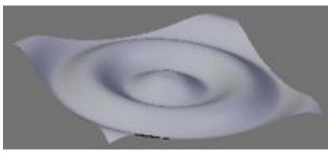
The Wave modifier adds an ocean−like motion to the Z coordinate of the Object Mesh.

Options

Wave Modifier Panel



Linear Wave Front



Circular Wave Front

This panel is found in the Editing buttons, Modifier panel, selecting Wave as the modifier.

X and Y

The Wave effect deforms vertices in the Z direction, and originates from the given starting point and propagates along the Mesh with circular wave fronts, or with rectilinear wave fronts, parallel to the X or Y axis. This is controlled by the two X and Y toggle buttons. If just one button is pressed fronts are linear, if both are pressed fronts are circular.

Cycl
: Repeats the waves cyclically, rather than a single pulse.

Normal
: Displaces the mesh along the surface normals.

Time sta
: The Frame at which the Wave begins (if Speed is positive), or ends (if Speed is negative). Use a negative frame number to prime and pre−start the waves.

Lifetime
: The number of frames in which the effect lasts. 0=forever, else frame at which waves stop being simulated.

Damptime
: An additional number of frames in which the wave slowly dampens from the Height value to zero after Lifetime is reached. The dampening occurs for all the ripples and begins in the first frame after the Lifetime is over. Ripples disappear over Damptime frames.

Sta X and Sta Y
: The starting points of the waves, in the Mesh object's Local coordinates.

Ob
: Use an Object as reference for starting position of the wave. Leave blank to disable.

VGroup
: Vertex group name with which to displace the Object.

Texture
: Affects the Object's displacement based on a texture.

Local
: This menu let's you choose the texture's coordinates for displacement.

Speed
: The speed, in Blender units per frame, of the ripple.

Height
: The height or amplitude, in Blender units, of the ripple.

Width
: Half of the width, in Blender units, between the tops of two subsequent ripples (if Cycl is enabled). This has an indirect effect on the ripple amplitude − if the pulses are too near to each other, the wave may not reach the z=0 position, so in this case Blender actually lowers the whole wave so that the minimum is zero and, consequently, the maximum is lower than the expected amplitude. See *Technical Details* below.

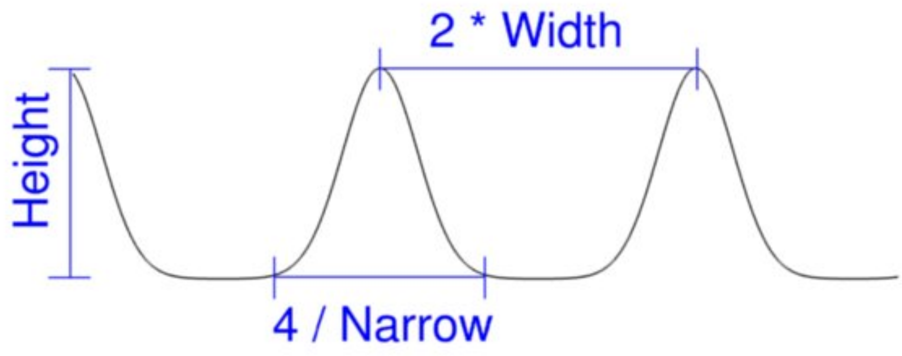
Narrow
: The actual width of each pulse, the higher the value the narrower the pulse. The actual width of the area in which the single pulse is apparent is given by 4 divided by the Narrow value. That is, if Narrow is 1 the pulse is 4 units wide, and if Narrow is 4 the pulse is 1 unit wide.

Warning

All the values described above must be multiplied with the corresponding Scale values of the object to get the real dimensions. For example, if the value of Scale Z: is **2** and the value of Height of the waves is **1**, it gives us final waves with a height of **2BU**!

Technical Details and Hints

The relationship of the above values is described here:



Wave front characteristics

To obtain a nice wave effect similar to sea waves and close to a sinusoidal wave, make the distance between following ripples and the ripple width equal, that is the Narrow value must be equal to 2 divided by the Width value. E.g. for *Width*=1 set *Narrow* to 2.

Previous: Manual/UVProject Modifier	Contents	*Next:* Manual/Lighting

User Manual: Contents | Guidelines | Blender Version **2.4x**

Introduction

Lighting is a very important topic in rendering, standing equal to modelling, materials and textures. The most accurately modelled and textured scene will yield poor results without a proper lighting scheme, while a simple model can become very realistic if skilfully lit. Lighting, sadly, is often overlooked by the inexperienced artist who commonly believes that, since real world scenes are often lit by a single light (a lamp, the sun, etc.) a single light will also do in computer graphics. This is false because in the real world even if a single light source is present, the light shed by such a source bounces off objects and is re−irradiated all over the scene, making shadows soft and shadowed regions not pitch black, but partially lit.

Viewing Restrictions

The color of an object and the lighting of your scene is affected by:

- Your ability to see different colors (partial color blindness is common)
- The medium in which you are viewing the image (e.g. an LCD panel versus printed glossy paper)
- The quality of the image (e.g. a JPEG at 0.4 compression versus 1.0)
- The environment in which you are viewing the image (e.g. a CRT Monitor with glare versus in a dark room, or in a sunshiny blue room)
- Your brain's perception of the color and intensity relative to those objects around it and the world background color.

So, the exact same image viewed by Person A on monitor B in room C may look very different to Person D viewing a printout E of the image while on the subway F.

Global Influences

In Blender, the things under your control that affect lighting are:

- The color of the world ambient light
- The use of Ambient Occlusion as a way to cast that ambient light onto the object
- The degree to which the ambient light colors the material of the object
- The use of Radiosity, where the color of one object radiates onto another
- The render engine used (Blender Internal versus Yafray)
- The lights in your scene

The physics of light bouncing around in the real−world is simulated by Ambient Occlusion (a world setting), buffer shadows (which approximate shadows being cast by objects), ray tracing (which traces the path of photons from a light source). Also, within Blender you can use the Radiosity engine. Ray tracing, ambient occlusion, and radiosity are compute−intensive processes. Blender can perform much faster rendering with its internal scan line renderer, which is a very good scan line renderer indeed. This kind of rendering engine is much faster since it does not try to simulate the real behavior of light, assuming many simplifying hypotheses.

Lighting Settings

Only after the above global influences do you get into adding on light from lamps in your scene. The main

things under your control are the:

- Type of light used (sun, spot, lamp, hemi, etc)
- Color of the light
- Position of the light and its direction
- Settings for each of those lights, including energy and dropoff

Then you are back to how that material's shader reacts to the light.

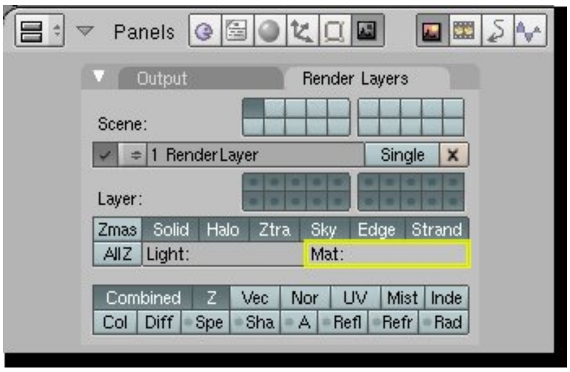
This chapter attempts to address the above, including how lights can work together in rigs to light your scene. In this chapter we will analyse the different type of lights in Blender and their behavior, we will analyze their strong and weak points. We also describe many lighting rigs, including the ever−popular three point light method.

Lighting in the Workflow

In this User Manual we have placed Lighting before Materials; you should set up your lighting before assigning materials to your meshes. Since the material shaders react to light, without proper lighting, the material shaders will not look right, and you will end up fighting the shader, when it is really the bad lighting that is causing you grief. All of the example images in this section do not use any material setting at all on the ball, cube or background.

Over−riding Materials to reset lighting

If you have started down the road of assigning materials, and are just now fiddling with the lighting, we suggest that you create a default, generic grey material; no VCol, no TexFace, no Shadeless, just plain old middle grey with an RGB of (0.8,0.8,0.8). If you click the auto−namer button, it should fill in "Grey". Next go to Scene context of the Buttons window and find the render buttons.



Mat: field highlighted in yellow.

You should see there the Render Layers panel. After you have found it, enter "Grey" into the Mat: field. If the name sticks, you know you entered it correctly. This will override any materials you may have set, and render everything with this flat boring color. Using this material, you can now go about adjusting the lighting.

Previous: Manual/Wave Modifier	Contents	*Next:* Manual/Lamp Types

User Manual: Contents | Guidelines | Blender Version **2.43**

Lamps

Mode: All Modes

Panel: Shading/Lamp Context Preview

Hotkey: Shift A to add new, F6 to change settings

Menu: Add Lamp

Description

The appearance of objects in a rendered image is determined by a few things, but mainly the objects' materials and light sources. Setting up lamps and lighting a scene is one of the most important parts of getting a good render, and Blender provides a range of different lamp types for different purposes.

Lamps are visualised in the 3D View using a solid dot, surrounding circle, and wire frame of dashed lines. Each light source has its own visualisation style, but there are some common indicators available across the different lamp types:

- Shadows

    If shadows are enabled an additional dashed circle is added around the inner, solid circle. This allows you to quickly see that this lamp has shadows enabled.

- Vertical Height Marker

Screenshot of a default Lamp, showing the Visual Height Marker.

A dim grey line, that helps locate the light relative to the global X−Y plane. The transparency of this line can be adjusted in the theme user preferences, with the Alpha value of the 3D View/Lamp item.

Options

Lamps can be added to a scene using the Add menu in the top header, or by using the add menu in the toolbox (SpaceAddLamp) or Shift A.

Lighting Groups

A material (and thus, all objects using that material) are lit (by default) by all lights in the visible layers, or only by a specific group of lights. To limit lighting only by a group of lights, you enter the name of the group in the GR: input field for the Material on the Shaders panel (this panel is also where you define how much Ambient light the object receives.)

Limit Lighting to Lamps in this Group highlighted in yellow.

Simply LMB this field, and type in the name of the lighting group. Of course, you must have previously defined this group of lights. Enable the Exclusive button if you want the lighting group to ONLY light objects with this specific material.

Other Options

If you look farther down the side of the shaders tab, you will see several other buttons.

- Cubic:

  Uses cubic interpolation of diffuse values for smoother transitions (between light and dark).

- Bias:

  Eliminates ray−traced shadow errors with the Phong specular shader.

Without Cubic enabled.

With Cubic enabled.

Animation switching between Non−Cubic and Cubic shadowing. You will need a modern, standards

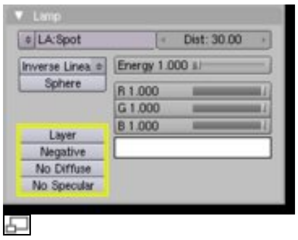
compliant, browser to see the animation. Click to View Animation.

Textures

In addition to putting out flat, uniform, light, you can assign a texture to affect the color of any lamp. Subtle textures add softness, while hard textures can simulate the shade from a tree. These textures are assigned to one of 10 channels, and behave exactly like material textures, except that they can only affect the color of the light. If the texture color is the same as the lamp color, you could say that the texture affects the *intensity* of the light.

What the Light Affects

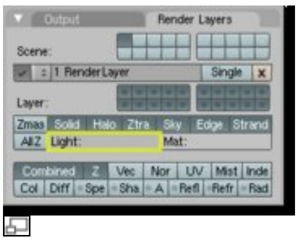
Every light has a common set of switches that, when enabled, control what the light affects:



Lamp panel with the Light affecting options highlighted in yellow.

- Layer: Only objects that are on the same layer as the light are lit by the light
- Negative: The light takes away light from the surface, subtracting light and making the surface darker, not lighter.
- No Diffuse: the light does not brighten the color of the surface
- No Specular: the light does not cause a shine on the surface, and is not used in calculating the Material Specular color or highlights on the surface

Object/Material
: Individual lights can also be put into a named group. If you enter the name of this group in the GR: field on the Material Shaders panel, then objects with that material will only be lit by lights in group.

Scene
: 

Screenshot showing the render layers panel with the Light: field highlighted in yellow. It is used to override which lights are used by the Render for rendering a scene.

This group name can also be entered in the Light: field in the Scene−>Render Layers tab, and then the whole scene will only be lit by the lights in that group.

Previous: Manual/Lighting	Contents	*Next:* Manual/Lamp

User Manual: Contents | Guidelines | Blender Version **2.46**

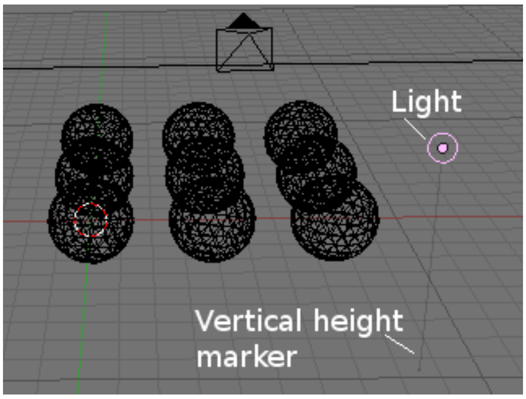
Lamp

Mode: All Modes

Panel: Shading/Lamp Context

Hotkey: F5

Description

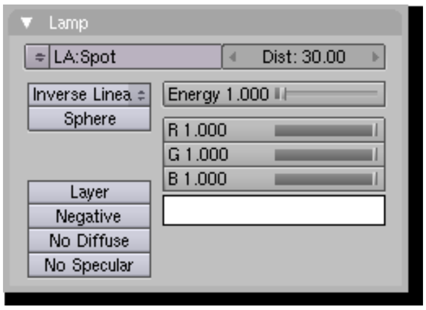


Lamp Light.

The 'Lamp' lamp is an omni−directional point of light, that is, a point radiating the same amount of light in all directions. It's visualised by a plain, circled dot, (*Lamp Light*).

Being a point light source, the direction of the light hitting an object's surface is determined by the line joining the lamp and the point on the surface of the object itself. Light intensity/energy decays based on (among other variables) distance from the Lamp to the object. In other words, surfaces that are further away are darker.

Options



Lamp Panel

- Dist: (Distance)

  The Dist: field indicates the number of Blender Units (BU) at which the intensity of the current light source will be half of its Intensity. Objects less than the number of BU away from the lamp will get

more light, while Objects further away will receive less light. Certain settings and lamp falloff types affect how the Dist: field is interpreted, meaning that it will not always react the same.

- Energy (0.0 − 10.0)

  The Intensity of the light sources illumination.

- Color

  The color of the light sources illumination.

- Layer

  Only objects that are on the same layer as the light are lit by the light.

- Negative

  The light takes away light from the surface, subtracting light and making the surface darker, not lighter.

- No Diffuse

  The light does not brighten the color of the surface.

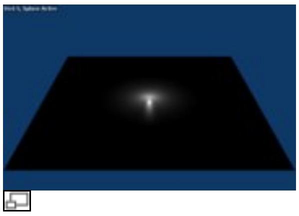
- No Specular

  The light does not cause a shine on the surface, and is not used in calculating the Material Specular color or highlights on the surface.

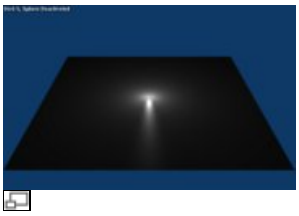
- Sphere

  The Sphere option restricts the lamp's illumination range so that it will instantly stop illuminating past an area once it reaches the number of Blender Units away from itself, as specified in the Dist: field.

  An imaginary Sphere (with a radius of the Dist: field) is placed around the light source and it's light is blocked from passing through the Sphere walls. So the Dist: field now basically means that any light that is further away form its light source than the value in the Dist: field will be Attenuated to 0 after this point, and won't naturally attenuate but instantly stop.

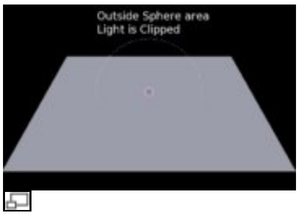


Screenshot showing the light Attenuation of a Constant Falloff light type with the Sphere option active.



Screenshot showing the light Attenuation of a Constant Falloff light type with the Sphere option Deactivated.

When the Sphere option is active, a dotted Sphere will appear around the light source, indicating the demarcation point at which this lights propagation will cease, example below:



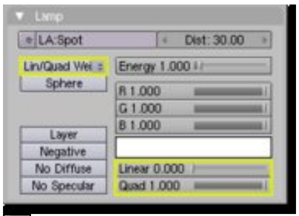
Screenshot of the 3D view window, showing the Sphere light clipping circle.

- Lamp Falloff

  The Lamp Falloff field has been improved in Blender 2.46 so the layout of the Lamp panel is different in a number of ways. One of the changes is the Lamp Falloff drop down menu. The Lamp Falloff types are listed and described below:

- Lin/Quad Weight

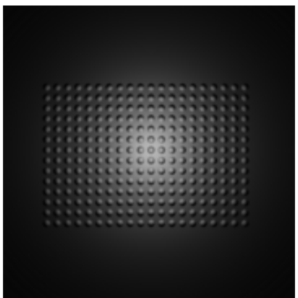
  This Lamp Falloff is described in the [Blender 2.46 release notes] as follows:
  "Exactly the same as in older Blenders with the old 'Quad' button enabled. When this setting is chosen, two sliders are shown, 'Linear' and 'Quad' (previously Quad1 and Quad2), which controls the 'linearness' or 'quadraticness' of the falloff curve. Lamps in old files with the 'Quad' button on will be initialised to this setting."



Lamp Panel with Lamp Falloff type Lin/Quad Weighted selected and the Quad and Linear slider also highlighted in yellow also.

So it looks as if the Lin/Quad Weighted Lamp Falloff type is in effect allowing the mixing of the 2 Light Attenuation profiles (Linear Attenuation type and Quadratic Attenuation type).

Here is a screenshot of the Lin/Quad Weighted light with default settings:



Screenshot showing the Lin/Quad Weighted Lamp Falloff type effect with default settings.

◊ Linear
This slider/numeric input field, can have a value between 0 and 1. A value of 1 in the Linear field and 0 in the Quad field, in effect means that the light from this source is completely Linear. Meaning that by the number of Blender Units distance specified in the Dist: field, this

light sources Intensity will be half the value it was when it reaches the number of Blender Units distance specified in the Dist: field.

In the situation just described the Linear Falloff type is being completely respected, it has the Intensity value it should have (half Intensity) by the time it reaches the distance specified in the Dist: field.

When the Quad slider is set to 0 the formula for working out the Attenuation at a particular range for Linear Attenuation is, in effect:

$I = E * (D / (D + Q_1 * R))$

Where E is the current Energy slider setting.
Where D is the current setting of the Dist: field.
Where Q_1 is the current setting of the Linear slider.
Where R is the distance from the lamp where the light Intensity gets measured.
Where I is the calculated Intensity of light.

◊ Quad
Quad (Quadratic) Attenuation type lighting is considered a more accurate representation of how light Attenuates, and as such when the Lin/Quad Weighted Lamp Fallout type is selected, Fully Quadratic Attenuation is selected by default (that is the Quad slider field is 1 and the Linear slider field is 0).

This slider/numeric input field, can have a value between 0 and 1. A value of 1 in the Quad field and 0 in the Linear field, in effect means that the light from this source is completely Quadratic (Quad type).

In the situation just described the Quad Falloff type is being completely respected so it has the Intensity value it should have (half intensity) by the time it reaches the distance specified in the Dist: field. After the light has reached the distance in the Dist: field, the light decays much more quickly.

One of the characteristics of Quadratic Light Attenuation is that at first it gradually Attenuates and then at a certain point starts to Attenuate at a much faster rate. The faster rate stage of Attenuation is roughly entered when the distance from the light is more than the value in the Dist: field.

When the Linear slider is set to 0 the formula for working out the attenuation at a particular range for Quadratic attenuation is, in effect:

$I = E * (D^2 / (D^2 + Q_2 * R^2))$

Where E is the current Energy slider setting.
Where D is the current setting of the Dist: field.
Where Q_2 is the current setting of the Quad slider.
Where R is the distance from the lamp where the light Intensity gets measured.
Where I is the calculated Intensity of light.

◊ Light Attenuation profile when both Linear and Quad sliders have values greater than 0
If both the Linear and Quad slider fields have values greater than 0, then the formula used to calculate the Light Attenuation profile changes to this:

$I = E * (D / (D + Q_1 * R)) * (D^2 / (D^2 + Q_2 * R^2))$

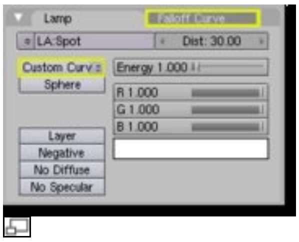
Where E is the current Energy slider setting.
Where D is the current setting of the Dist: field.
Where Q_1 is the current setting of the Linear slider.
Where Q_2 is the current setting of the Quad slider.
Where R is the distance from the lamp where the light Intensity gets measured.
Where I is the calculated Intensity of light.

◊ No Light Attenuation when both Linear and Quad sliders have values of 0.
If both the Linear and Quad sliders have 0 as their values. Then their light Intensity will not Attenuate with distance. This does not mean that the light will not get darker, it will, but only because the Energy the light has is spread out over a wider and wider distance. The total amount of Energy in the spread out light will remain the same though. Light angle also affects the amount of light you see. If what you want is a light source that doesn't attenuate and gives the same amount of light Intensity to each area it hits you need a light with properties like the Constant Lamp Falloff type.

Also when the Linear and Quad sliders are both 0 values the Dist: field ceases to have any visible effect on the Light Attenuation.

- Custom Curve

The Custom Curve Lamp Falloff type became available in Blender 2.46 and is very flexible.



Lamp Panel with Lamp Falloff type Custom Curve selected and highlighted in Yellow. Also shown is the Falloff Curve tab that is created (also highlighted in Yellow) when Custom Curve falloff type is in effect.

Most other Lamp Falloff types work by having their light Intensity start at its maximum (when nearest to the Light source) and then with some predetermined pattern decrease their light Intensity when the Distance from the light source gets further away.

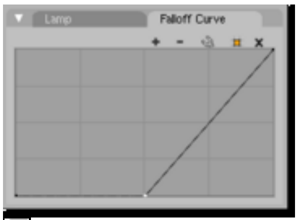
When using the Custom Curve Lamp Falloff type, a new panel is created called "Falloff Curve" shown below:

Falloff Curve panel used to control the amount of Light Attenuation a light has in an arbitrary manner.

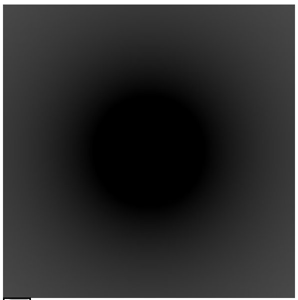
This Falloff Curve Profile Graph, allows the user to alter how Intense light is at a particular point along a lights Attenuation Profile.

In the example above (the default for the Falloff Curve Profile Graph), the Graph shows that the Intensity of the light starts off at the maximum Intensity that the light can have (when near the light) and linearly Attenuates the light Intensity as it moves to the right (further away from the light source).

So if the user wanted to have a Light Attenuation Profile that got more Intense as it moved away from the light source, the user could alter the Light Attenuation Profile Graph as needed. Below is an example of a Falloff Curve Profile Graph, showing just such a situation:



Falloff Curve for reversed Attenuation.



Falloff Curve for reversed Attenuation rendered.

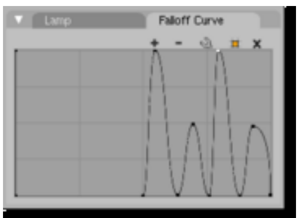
You are not just limited to simple changes such as light reversing the Attenuation profile, you can have almost any Attenuation profile you desire.

The Falloff Curve Profile Graph has 2 axis, the Intensity axis and the Distance axis, labelled Intensity and Distance in the pictures shown (although these labels were added to make describing how the Falloff Curve Profile Graph works, easier, they don't appear in Blender).

The Distance axis represents the position at a particular point along a light sources Attenuation path. The far left being at the the position of the light source and the far right being the place where the light sources influence would normally be completely Attenuated. I say normally would because the Falloff Curve can be altered to do the exact opposite if required.

The Intensity axis represents the Intensity at a particular point along a light sources Attenuation path. Higher Intensity is represented by being higher up the Intensity axis while lower Intensity light is represented by being lower down on the Intensity axis.

Here is another example of different Falloff Curve Profile Graph, along with its resultant render output:



Falloff Curve Profile Graph resulting in Oscillating Attenuation pattern in Light.

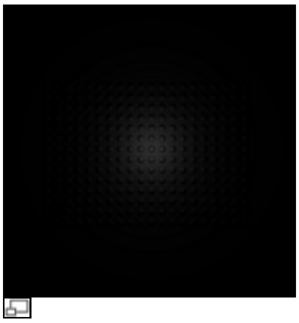
Render showing the affect of the Falloff Curve Profile Graph on the Attenuation.

Altering the Falloff Curve Profile Graph is easy. Just LMB on a part of the graph you want to alter and drag it where you want it to be. If when you click you are over or near one of the tiny black

square handles, it will turn white indicating that this is the handle that is now selected and you will be able to drag it to a new position. If when you click on the graph you are not near a handle, one will be created at the point that you clicked, which you can then drag where you wish.

- Inverse Square

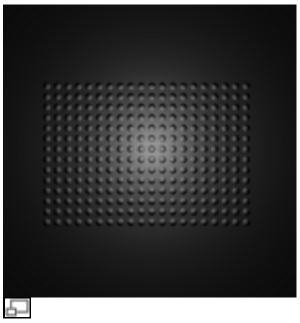
This Lamp Fallout type Attenuates its Intensity according to inverse square law, scaled by the 'Dist:' value. Inverse square is a sharper, realistic decay, useful for lighting such as desk lamps and street lights. This is similar to the old Quad option with slight changes.



Screenshot showing the Inverse Square Lamp Falloff type effect with default settings.

- Inverse Linear

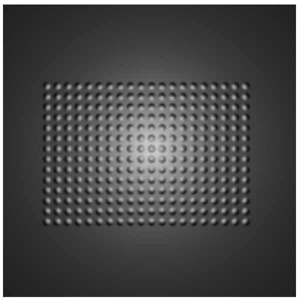
This Lamp Fallout type Attenuates its Intensity linearly, scaled by the 'Dist' value. This is the default setting, behaving the same as the default in previous Blender versions without 'Quad' switched on. This isn't physically accurate, but can be easier to light with.



Screenshot showing the Inverse Linear Lamp Falloff type effect with default settings.

- Constant

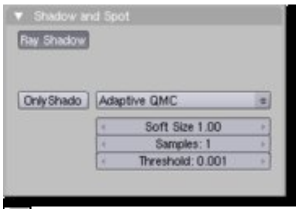
This Lamp Fallout type does not Attenuate its Intensity with distance. This is useful for distant light sources like the sun or sky, which are so far away that their falloff isn't noticeable. Sun and Hemi lamps always have constant falloff.



Screenshot showing the Constant Lamp Falloff type effect with default settings.

Shadow and Spot Panel for Lamp and Sun light sources

When a Lamp or Sun light source are selected the Spot and Shadow Panel has the following default layout:



The Shadow and Spot Panel when Lamp or Sun light sources are selected.

Options

- Ray Shadow

  The Ray Shadow button enables the Lamp and Sun light sources to generate Ray Traced Shadows.

  When the Ray Shadow button is selected, another set of options is made available, those options being:

  ◊ Shadow Sample Generator Type − Constant QMC

    The Constant QMC method is used to calculate shadow values in a very uniform, evenly distributed way. This method results in very good calculation of shadow value but it is not as fast as using the Adaptive QMC method, however Constant GMC is more accurate.

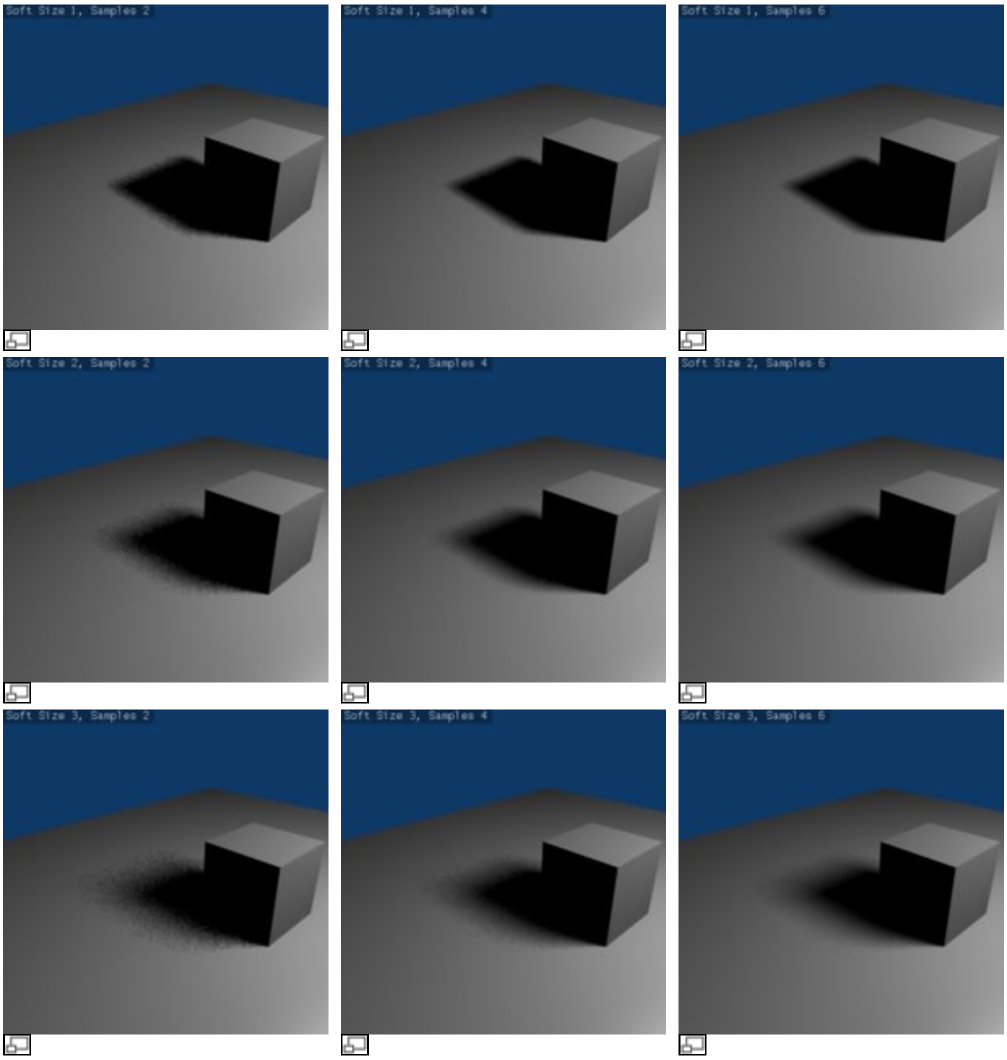
  ◊ Shadow Sample Generator Type − Adaptive QMC

    The Adaptive QMC method is used to calculate shadow values in a slightly less uniform and distributed way. This method results in good calculation of shadow value but not as good as Constant QMC. The advantage of using Adaptive QMC is that it in general is much quicker while being not much worse than Constant QMC in terms of overall results.

  ◊ Samples
    This Numerical slider field set the maximum number of samples that both Constant QMC and Adaptive QMC will use to do their shadow calculations. The maximum number of samples that can be taken is 16. According to the tooltip information that appears when over this field the sample value is squared so setting a sample value of 3 really means 3^2 samples will be taken.

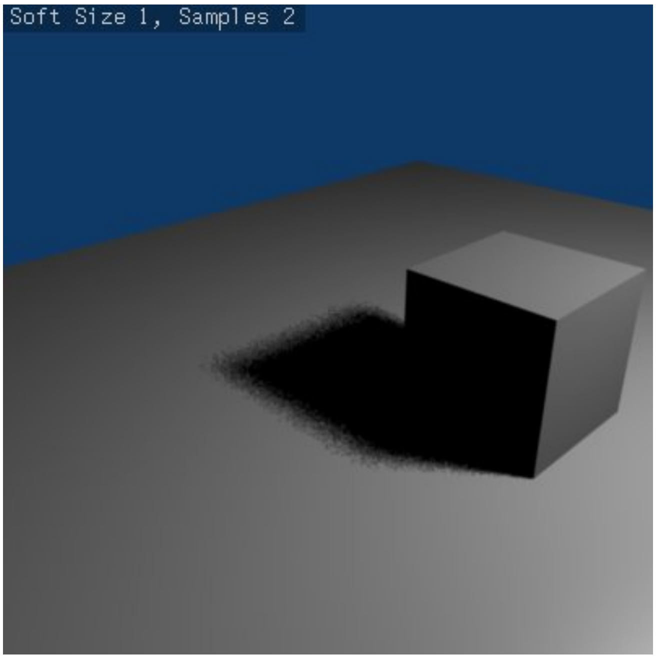
  ◊ Soft Size
    The Soft Size numeric slider, determines the size of the fuzzy/diffuse/penumbra area around the edge of a shadow. Soft Size only determines the width of the soft shadow size not how graduated and smooth the shadow is. If you want a wide shadow which is also soft and finely graduated you must also set the number of Samples in the Samples field higher than 1,

otherwise this field has no visible effect and the shadows generated will not have a soft edge. The maximum value for Soft Size is 100 (Blender Units?).



Above is a table of renders with different Soft Size and Sample settings showing the effect of various values on the softness of shadow edges.

Below is an Animated version of the above table of images showing the effects:



You may need to click on the Image to see the Animation.

◊ Threshold
The Threshold field is used with the Adaptive GMC shadow calculation method. The value in the Threshold field is used to determine if Adaptive GMC shadow sample calculation can skipped based on a threshold of how shadowed an area is already. The maximum Threshold value is 1.

• Only Shadow

When the Only Shadow button is selected the light source will not illuminate an object but will generate the shadows that would normally appear.

This feature is often used to control how and where shadows fall by having a light which illuminates but has no shadow, combined with a second light which doesn't illuminate but has Only Shadow enabled, allowing the user to control shadow placement by moving the Shadow Only light around.

What is Quasi−Monte Carlo?

The Monte Carlo method is a method of taking a series of samples/readings of values (any kind of values, such as light values, color values, reflective states) in or around an area at random, so as to determine the correct actions to take in certain calculations which usually require multiple sample values to determine overall accuracy, of those calculations. The Monte Carlo methods tries to be as random as possible, this can often cause areas that are being sampled to have large irregular gaps in them (places that are not sampled/read), this in turn can cause problems for certain calculations (such as shadow calculation).

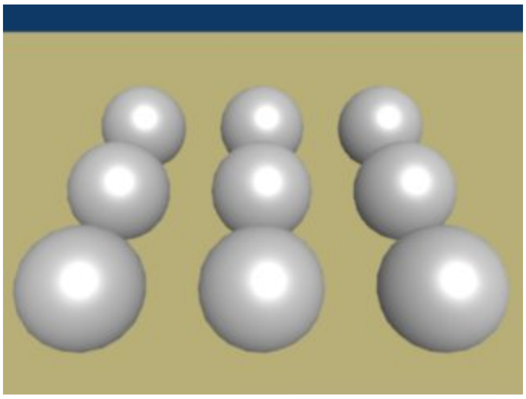
The solution to this was the Quasi−Monte Carlo method.

The Quasi−Monte Carlo method is also random, but tries to make sure that the samples/readings it takes are also better distributed (leaving less irregular gaps in its sample areas) and more evenly spread across an area. This has the advantage of sometimes leading to more accurate calculations based on samples/reading.

Note

Everything above this point it an update to reflect the state of things as of Blender 2.46, it could all be wrong or out of date for Blender 2.47, if you want me to change it let me know and I will, or edit it yourself. −− Terrywallwork −− 6th September 2008

Examples

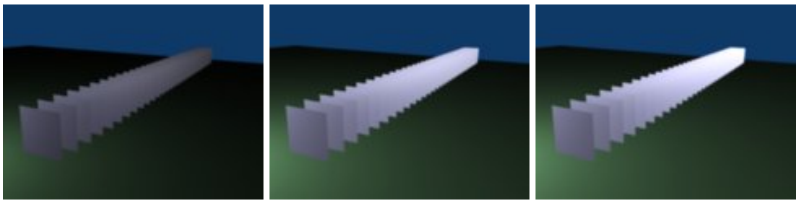


Render example.

Notice in (*Render example*) that the light is gradually diminishing for each sphere that is farther away from the light source as compared to the Sun render example where the light's intensity was constant (i.e. never faded with distance).

Distance

In this example, the Lamp has been set pretty close to the group of planes. This causes the light to affect the front, middle and rear planes more dramatically. Looking at (*Various Distance settings*) you can see that as the Dist is increased more and more objects are becoming progressively brighter.



Dist: 10 Dist: 100 Dist: 1000

"*Various Distance settings*", Shadow disabled.

The Dist: parameter is controlling where the light is falling −−at a linear rate−− to 1/2 its original value from the light's origin. As you increase or decrease this value you are changing where this 1/2 falloff occurs. You could think of Dist: as the surface of a sphere and the surface is where the light's intensity has fallen to 1/2 its strength, in all directions. Note that the light's intensity continues to fall even after Dist:. Dist: just specifies the distance where 1/2 of the light's energy has weakened.

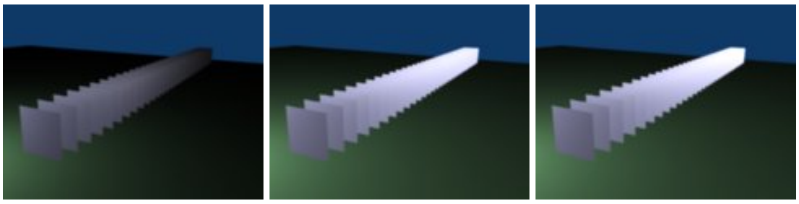
Notice in (*Dist: 1000*) that the farthest objects are very bright. This is because the falloff has been extended far into the distance which means the light is very strong when it hits the last few objects. It is not until 1000 units that the light's intensity has fallen to 1/2 its original intensity.

Contrast this with (*Dist: 10*) where the falloff occurs so soon that the farther objects are barely lit. The light's intensity has fallen by 1/2 by time it even reaches the 10th object.

You may be wondering, why the first few planes appear to be dimmer? This is because the surface angle between the light and the object's surface normal are getting close to oblique. That is the nature of a Lamp light object. By moving the light infinitely far away you would begin to approach the characteristics of the Sun lamp type.

Quad

Quad makes the light's intensity falloff with a non−linear rate, or specifically, quadratic rate. The characteristic feature of using "**Quad**" is that the light's intensity begins to fall off very slowly but then starts falling off very rapidly. We can see this in the (*Quad enabled*) images.



Quad with 10 Quad with 100 Quad with 1000

Quad enabled with the specified distances.

With Quad enabled the "**Dist:**" field is specifying where the light begins to fall faster, roughly speaking, see Technical Details for more info.

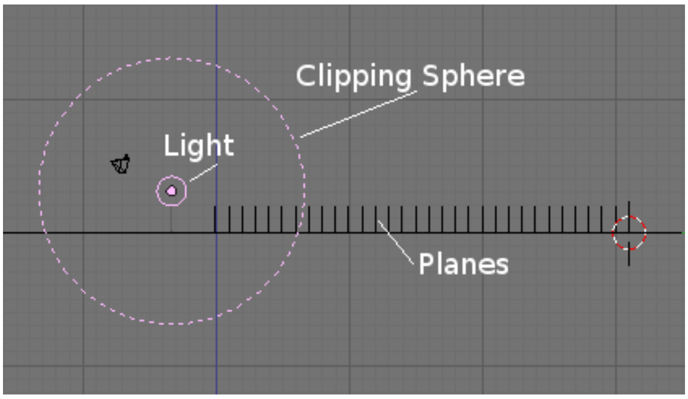
In (*Quad with 10*) the light's intensity has fallen so quickly that the last few objects aren't even lit.

Both (*Quad with 100*) and (*Quad with 1000*) appear to be almost identical and that is because the Distance is set beyond the farthest object's distance which is at ~40 units out. Hence, all the objects get almost the full intensity of the light.

As with Dist the first few objects are dimmer than farther objects because they are very close to the light. Remember, the brightness of an object's surface is also based on the angle between the surface normal of an object and the ray of light coming from the lamp.

This means there are at least two things that are controlling the surface's brightness: intensity and the angle between the light source and the surface's normal.

Sphere



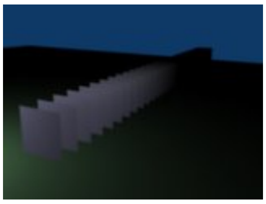
Clipping Sphere

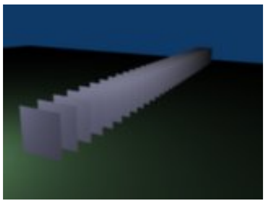
Sphere controls where the light's intensity is clipped/clamped off. All light rays stop at the surface of the sphere regardless of the light's falloff. In (*Clipping Sphere*) you can see a side view example of the setup with Sphere enabled and a distance of 10.

Any objects beyond the sphere receive no light from the lamp.

The Dist: field is now specifying both where the light's rays stop and the intensity's ratio falloff setting. The only difference is that now light abruptly stops at sphere's surface regardless of the light's intensity.


Sphere with 10


Sphere with 20


Sphere with 40

Sphere enabled with the specified distances, Quad disabled.

In (*Sphere with 10*) the clipping sphere's radius is 10 units which means the light's intensity is also being controlled by 10 units of distance. With Quad disabled the light's intensity has fallen very low even before it gets to the first object.

In (*Sphere with 20*) the clipping sphere's radius is now 20 units and some light is reaching the middle objects, but no light is going beyond the clipping sphere even if the light still has energy left.

In (*Sphere with 40*) the clipping sphere's radius is now 40 units which is beyond the last object. However, the light doesn't make it to the last few objects because the intensity has fallen to **0**. The light's intensity has faded before it was clipped by the sphere.

Hints

If the Lamp light is set to not cast shadows it illuminates through walls and the like. If you want to achieve some nice effects like a fire, or a candle−lit room interior seen from outside a window, the Sphere option is a must. By carefully working on the Distance value you can make your warm firelight shed only within the room, while illuminating outside with a cool moonlight, the latter achieved with a Sun or Hemi light or both.

Technical Details

The effect of the Distance parameter is very evident, while the effect of the Quad button is more subtle. In any case the absence of shadows is still a major issue. As a matter of fact only the first plane should be lit, because all the others should fall in the shadow of the first.

For the Math enthusiasts, and for those desiring deeper insight, the laws governing the decay are the following. Let D be the value of the Distance Numeric Button, E the value of the Energy slider and r the distance from the Lamp to the point where the light intensity I is to be computed.

If Quad and Sphere buttons are off:

$I = E \times (D / (D + r))$

It is evident what affirmed before: That the light intensity equals half the energy for $r = D$.

If Quad Button is on:

$I = E \times (D / (D + Q_1 r)) \times (D^2 / (D^2 + Q_2 r^2))$

This is a little more complex and depends from the Quad1 (Q_1) and Quad2 (Q_2) slider values. Nevertheless it is apparent how the decay is fully linear for

$Q_1 = 1, Q_2 = 0$

and fully quadratic for

$Q_1 = 0, Q_2 = 1$

this latter being the default. Interestingly enough if

$Q_1 = Q_2 = 0$

then light intensity does not decay at all. If the Sphere button is on the above computed light intensity *I* is further modified by multiplication by the term which has a linear progression for *r* from 0 to *D* and is identically 0 otherwise.

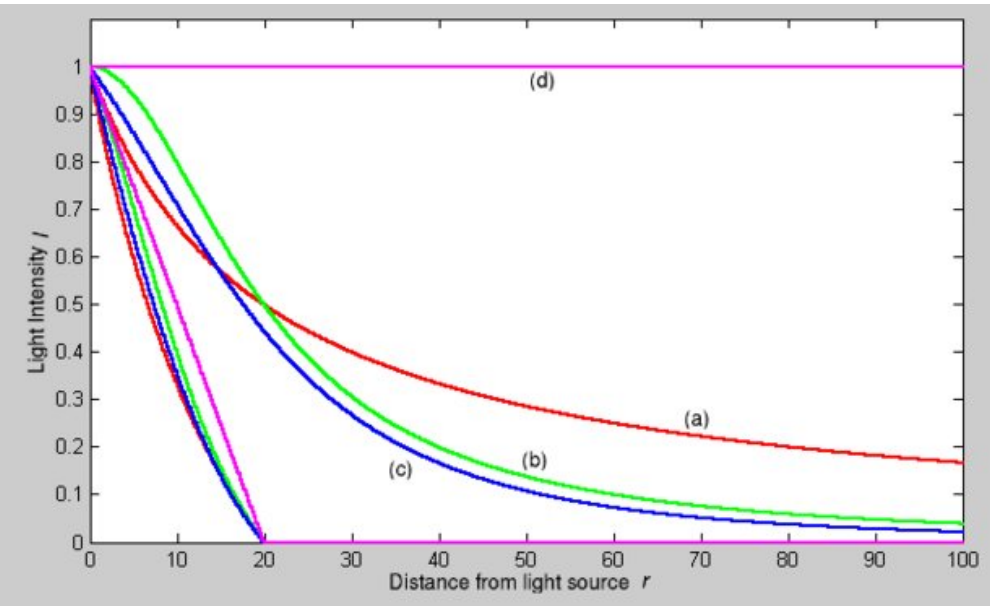
If the Quad button is off and the Sphere button is on:

$Is = E \times (D / (D + r)) \times ((D - r) / D)$ *if* $r < D$*; 0 otherwise*

If both Quad and Sphere buttons are on:

$Is = E \times (D / (D + Q_1 r)) \times (D^2 / (D^2 + Q_2 r^2)) \times ((D - r) / D)$ *if* $r < D$*; 0 otherwise*

Might be helpful in understanding these behaviours graphically.



Light decays: a) Blender default linear; b) Blender default quadratic with Quad1=0, Quad2=1; c) Blender quadratic with Quad1=Quad2=0.5; d) Blender quadratic with Quad1=Quad2=0. Also shown in the graph the

same curves, in the same colours, but with the Sphere button turned on.

See Also

Place related links here.

Previous: Manual/Lamp Types Contents *Next:* Manual/Spot Lamp

User Manual: Contents | Guidelines | Blender Version **2.48**

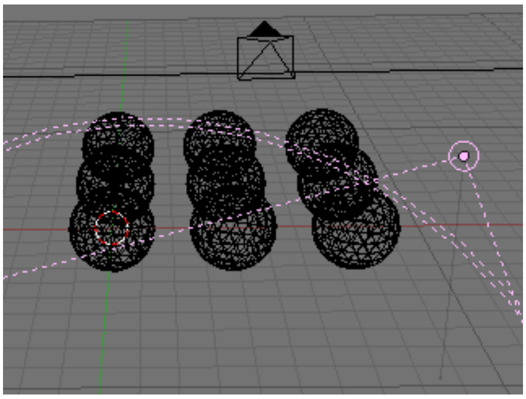
Spot Lamp

Mode: All Modes

Panel: Shading/Lamp Context

Hotkey: F5

Description

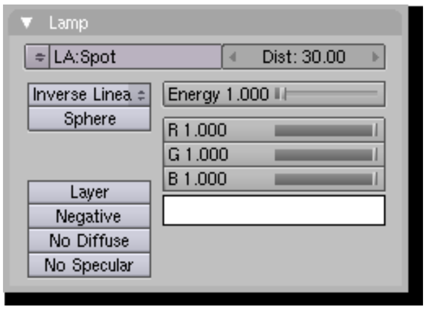


Spot Lamp

A Spot lamp emits a cone shaped beam of light from the tip of the cone, in a given direction.

The Spot light is the most complex of the light objects and indeed, for a long time, among the most used thanks to the fact that it was the only one able to cast shadows. Today, with the integration of a ray tracer within the internal render engine of Blender, all lamps can cast shadows (except Hemi). Even so, Spot lamps' shadow buffers are much faster to render than raytraced shadows, especially when blurred/softened, and spot lamps also provide other functionality such as 'volumetric' halos.

Options

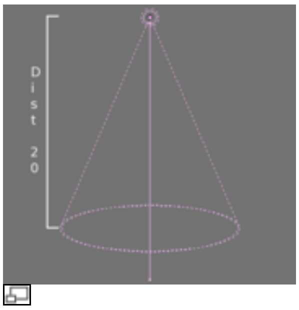


Lamp Panel

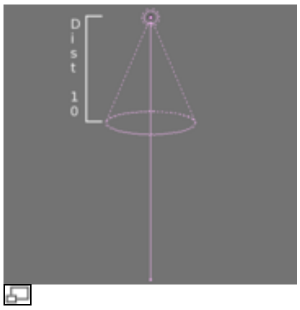
- Dist: (Distance)

The Dist: field indicates the number of Blender Units (BU) at which the intensity of the current light source will be half of its Intensity. Objects less than the number of BU away from the lamp will get more light, while Objects further away will receive less light. Certain settings and lamp falloff types affect how the Dist: field is interpreted, meaning that it will not always react the same.

Changing the Dist: field value when using a Spotlight also changes the appearance of the Spotlight as displayed in the 3D Viewport.



Spotlight with Dist value of 20



Spotlight with Dist value of 10

- Energy (0.0 − 10.0)

  The Intensity of the light sources illumination.

- Color

  The color of the light sources illumination.

- Layer

  Only objects that are on the same layer as the light are lit by the light.

- Negative

  The light takes away light from the surface, subtracting light and making the surface darker, not lighter.

- No Diffuse

  The light does not brighten the color of the surface.

- No Specular

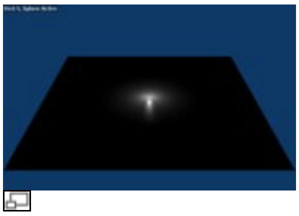
  The light does not cause a shine on the surface, and is not used in calculating the Material Specular color or highlights on the surface.

- Sphere

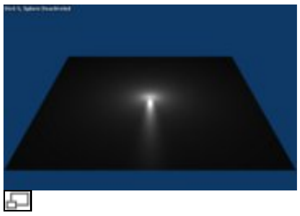
  The Sphere option restricts the Spotlights illumination range so that it will instantly stop illuminating past an area once it reaches the number of Blender Units away from itself, as specified in the Dist: field.

  An imaginary Sphere (with a radius of the Dist: field) is placed around the Spotlight source and it's light is blocked from passing through the Sphere walls. So the Dist: field now basically means that any light that is further away form its light source than the value in the Dist: field will be Attenuated

to 0 after this point, and won't naturally attenuate but instantly stop.



Screenshot showing the light Attenuation of a Constant Falloff light type with the Sphere option active.



Screenshot showing the light Attenuation of a Constant Falloff light type with the Sphere option Deactivated.

When the Sphere option is active, an imaginary Sphere will be projected around the light source, indicating the demarcation point at which this lights propagation will cease, example below:

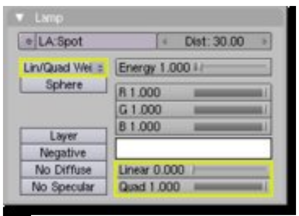
Screenshot of the 3D view window, showing the Sphere light clipping Sphere when using a Spotlight.

- Lamp Falloff

The Lamp Falloff field has been improved in Blender 2.46 so the layout of the Lamp panel is different in a number of ways. One of the changes is the Lamp Falloff drop down menu. The Lamp Falloff types are listed and described below:

- Lin/Quad Weight

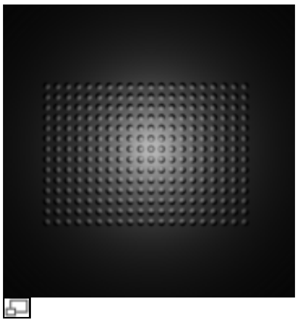
This Lamp Falloff is described in the Blender 2.46 release notes as follows:
"Exactly the same as in older Blenders with the old 'Quad' button enabled. When this setting is chosen, two sliders are shown, 'Linear' and 'Quad' (previously Quad1 and Quad2), which controls the 'linearness' or 'quadraticness' of the falloff curve. Lamps in old files with the 'Quad' button on will be initialised to this setting."



Lamp Panel with Lamp Falloff type Lin/Quad Weighted selected and the Quad and Linear slider also highlighted in yellow also.

So it looks as if the Lin/Quad Weighted Lamp Falloff type is in effect allowing the mixing of the 2 Light Attenuation profiles (Linear Attenuation type and Quadratic Attenuation type).

Here is a screenshot of the Lin/Quad Weighted light with default settings:



Screenshot showing the Lin/Quad Weighted Lamp Falloff type effect with default settings.

◊ Linear
This slider/numeric input field, can have a value between 0 and 1. A value of 1 in the Linear field and 0 in the Quad field, in effect means that the light from this source is completely Linear. Meaning that by the number of Blender Units distance specified in the Dist: field, this light sources Intensity will be half the value it was when it reaches the number of Blender Units distance specified in the Dist: field.

In the situation just described the Linear Falloff type is being completely respected, it has the Intensity value it should have (half Intensity) by the time it reaches the distance specified in the Dist: field.

When the Quad slider is set to 0 the formula for working out the Attenuation at a particular range for Linear Attenuation is, in effect:

$I = E * (D / (D + Q_1 * R))$

Where E is the current Energy slider setting.
Where D is the current setting of the Dist: field.
Where Q_1 is the current setting of the Linear slider.
Where R is the distance from the lamp where the light Intensity gets measured.
Where I is the calculated Intensity of light.

◊ Quad
Quad (Quadratic) Attenuation type lighting is considered a more accurate representation of how light Attenuates, and as such when the Lin/Quad Weighted Lamp Fallout type is selected, Fully Quadratic Attenuation is selected by default (that is the Quad slider field is 1 and the Linear slider field is 0).

This slider/numeric input field, can have a value between 0 and 1. A value of 1 in the Quad field and 0 in the Linear field, in effect means that the light from this source is completely Quadratic (Quad type).

In the situation just described the Quad Falloff type is being completely respected so it has the Intensity value it should have (half intensity) by the time it reaches the distance specified in the Dist: field. After the light has reached the distance in the Dist: field, the light decays much more quickly.

One of the characteristics of Quadratic Light Attenuation is that at first it gradually Attenuates and then at a certain point starts to Attenuate at a much faster rate. The faster rate stage of Attenuation is roughly entered when the distance from the light is more than the value in the Dist: field.

When the Linear slider is set to 0 the formula for working out the attenuation at a particular range for Quadratic attenuation is, in effect:

$I = E * (D^2 / (D^2 + Q_2 * R^2))$

Where E is the current Energy slider setting.
Where D is the current setting of the Dist: field.
Where Q_2 is the current setting of the Quad slider.
Where R is the distance from the lamp where the light Intensity gets measured.
Where I is the calculated Intensity of light.

◊ Light Attenuation profile when both Linear and Quad sliders have values greater than 0
If both the Linear and Quad slider fields have values greater than 0, then the formula used to calculate the Light Attenuation profile changes to this:

$I = E * (D / (D + Q_1 * R)) * (D^2 / (D^2 + Q_2 * R^2))$

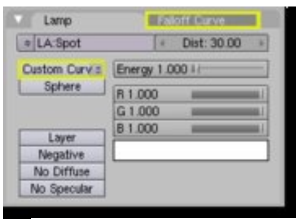
Where E is the current Energy slider setting.
Where D is the current setting of the Dist: field.
Where Q_1 is the current setting of the Linear slider.
Where Q_2 is the current setting of the Quad slider.
Where R is the distance from the lamp where the light Intensity gets measured.
Where I is the calculated Intensity of light.

◊ No Light Attenuation when both Linear and Quad sliders have values of 0.
If both the Linear and Quad sliders have 0 as their values. Then their light Intensity will not Attenuate with distance. This does not mean that the light will not get darker, it will, but only because the Energy the light has is spread out over a wider and wider distance. The total amount of Energy in the spread out light will remain the same though. Light angle also affects the amount of light you see. If what you want is a light source that doesn't attenuate and gives the same amount of light Intensity to each area it hits you need a light with properties like the Constant Lamp Falloff type.

Also when the Linear and Quad sliders are both 0 values the Dist: field ceases to have any visible effect on the Light Attenuation.

- Custom Curve

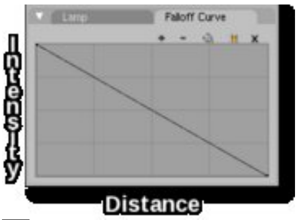
The Custom Curve Lamp Falloff type became available in Blender 2.46 and is very flexible.



Lamp Panel with Lamp Falloff type Custom Curve selected and highlighted in Yellow. Also shown is the Falloff Curve tab that is created (also highlighted in Yellow) when Custom Curve falloff type is in effect.

Most other Lamp Falloff types work by having their light Intensity start at its maximum (when nearest to the Light source) and then with some predetermined pattern decrease their light Intensity when the Distance from the light source gets further away.

When using the Custom Curve Lamp Falloff type, a new panel is created called "Falloff Curve" shown below:

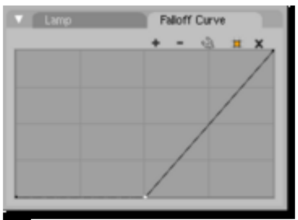


Falloff Curve panel used to control the amount of Light Attenuation a light has in an arbitrary manner.

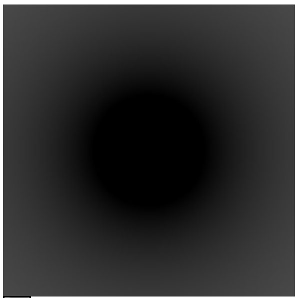
This Falloff Curve Profile Graph, allows the user to alter how Intense light is at a particular point along a lights Attenuation Profile.

In the example above (the default for the Falloff Curve Profile Graph), the Graph shows that the Intensity of the light starts off at the maximum Intensity that the light can have (when near the light) and linearly Attenuates the light Intensity as it moves to the right (further away from the light source).

So if the user wanted to have a Light Attenuation Profile that got more Intense as it moved away from the light source, the user could alter the Light Attenuation Profile Graph as needed. Below is an example of a Falloff Curve Profile Graph, showing just such a situation:



Falloff Curve for reversed Attenuation.



Falloff Curve for reversed Attenuation rendered.

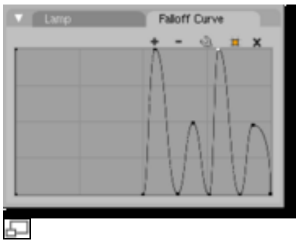
You are not just limited to simple changes such as light reversing the Attenuation profile, you can have almost any Attenuation profile you desire.

The Falloff Curve Profile Graph has 2 axis, the Intensity axis and the Distance axis, labelled Intensity and Distance in the pictures shown (although these labels were added to make describing how the Falloff Curve Profile Graph works, easier, they don't appear in Blender).

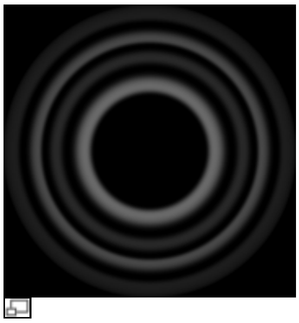
The Distance axis represents the position at a particular point along a light sources Attenuation path. The far left being at the the position of the light source and the far right being the place where the light sources influence would normally be completely Attenuated. I say normally would because the Falloff Curve can be altered to do the exact opposite if required.

The Intensity axis represents the Intensity at a particular point along a light sources Attenuation path. Higher Intensity is represented by being higher up the Intensity axis while lower Intensity light is represented by being lower down on the Intensity axis.

Here is another example of different Falloff Curve Profile Graph, along with its resultant render output:



Falloff Curve Profile Graph resulting in Oscillating Attenuation pattern in Light.

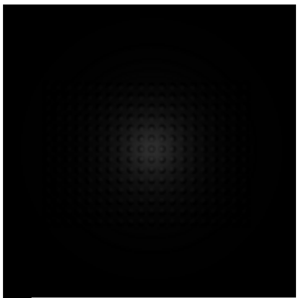


Render showing the affect of the Falloff Curve Profile Graph on the Attenuation.

Altering the Falloff Curve Profile Graph is easy. Just LMB on a part of the graph you want to alter and drag it where you want it to be. If when you click you are over or near one of the tiny black square handles, it will turn white indicating that this is the handle that is now selected and you will be able to drag it to a new position. If when you click on the graph you are not near a handle, one will be created at the point that you clicked, which you can then drag where you wish.

- Inverse Square

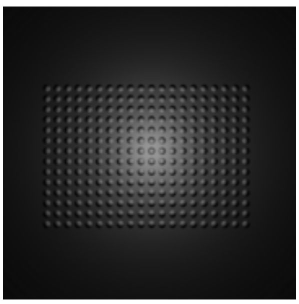
This Lamp Fallout type Attenuates its Intensity according to inverse square law, scaled by the 'Dist:' value. Inverse square is a sharper, realistic decay, useful for lighting such as desk lamps and street lights. This is similar to the old Quad option with slight changes.



Screenshot showing the Inverse Square Lamp Falloff type effect with default settings.

- Inverse Linear

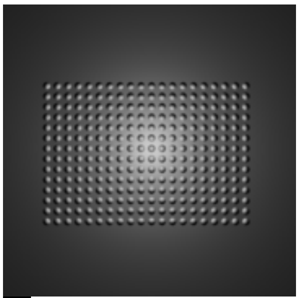
This Lamp Fallout type Attenuates its Intensity linearly, scaled by the 'Dist' value. This is the default setting, behaving the same as the default in previous Blender versions without 'Quad' switched on. This isn't physically accurate, but can be easier to light with.



Screenshot showing the Inverse Linear Lamp Falloff type effect with default settings.

- Constant

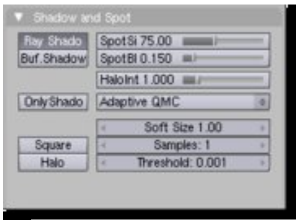
This Lamp Fallout type does not Attenuate its Intensity with distance. This is useful for distant light sources like the sun or sky, which are so far away that their falloff isn't noticeable. Sun and Hemi lamps always have constant falloff.



Screenshot showing the Constant Lamp Falloff type effect with default settings.

Shadow & Spot Panel for a Spotlight lighting type

When a Spolight lighting type is selected the following default layout for the Shadow & Spot Panel is shown:



The Shadow and Spot Panel when Spoltlight lighting is are selected.

Options

- Ray Shadow

The Ray Shadow button enables the Lamp and Sun light sources to generate Ray Traced Shadows.

When the Ray Shadow button is selected, another set of options is made available, those options being:

◊ Shadow Sample Generator Type − Constant QMC

The Constant QMC method is used to calculate shadow values in a very uniform, evenly distributed way. This method results in very good calculation of shadow value but it is not as fast as using the Adaptive QMC method, however Constant GMC is more accurate.

◊ Shadow Sample Generator Type − Adaptive QMC

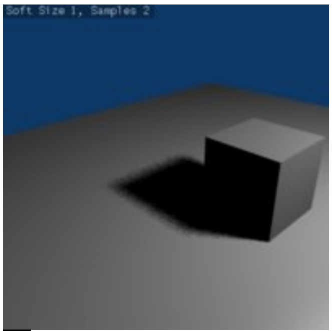
The Adaptive QMC method is used to calculate shadow values in a slightly less uniform and distributed way. This method results in good calculation of shadow value but not as good as Constant QMC. The advantage of using Adaptive QMC is that it in general is much quicker while being not much worse than Constant QMC in terms of overall results.

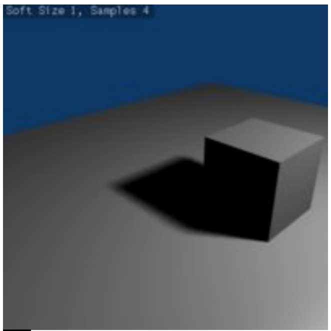
◊ Samples

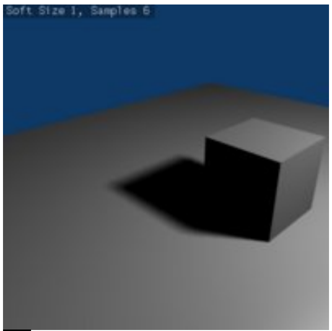
This Numerical slider field set the maximum number of samples that both Constant QMC and Adaptive QMC will use to do their shadow calculations. The maximum number of samples that can be taken is 16. According to the tooltip information that appears when over this field the sample value is squared so setting a sample value of 3 really means 3^2 samples will be taken.

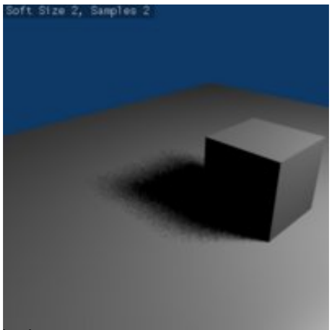
◊ Soft Size

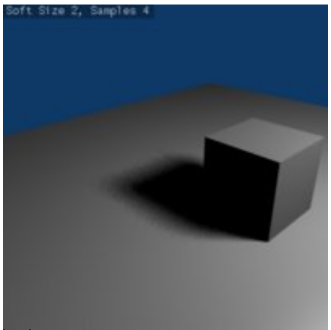
The Soft Size numeric slider, determines the size of the fuzzy/diffuse/penumbra area around the edge of a shadow. Soft Size only determines the width of the soft shadow size not how graduated and smooth the shadow is. If you want a wide shadow which is also soft and finely graduated you must also set the number of Samples in the Samples field higher than 1, otherwise this field has no visible effect and the shadows generated will not have a soft edge. The maximum value for Soft Size is 100 (Blender Units?).

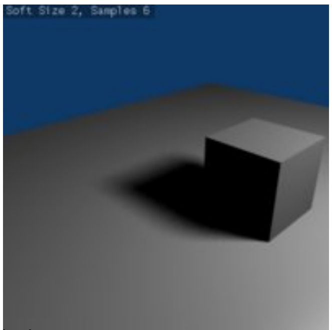


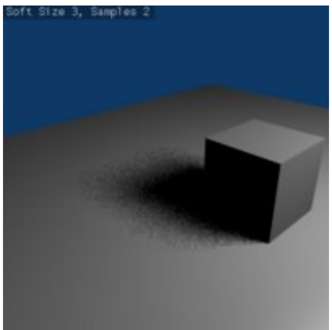


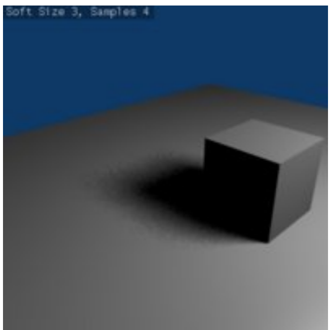


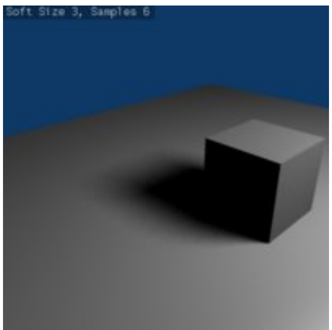








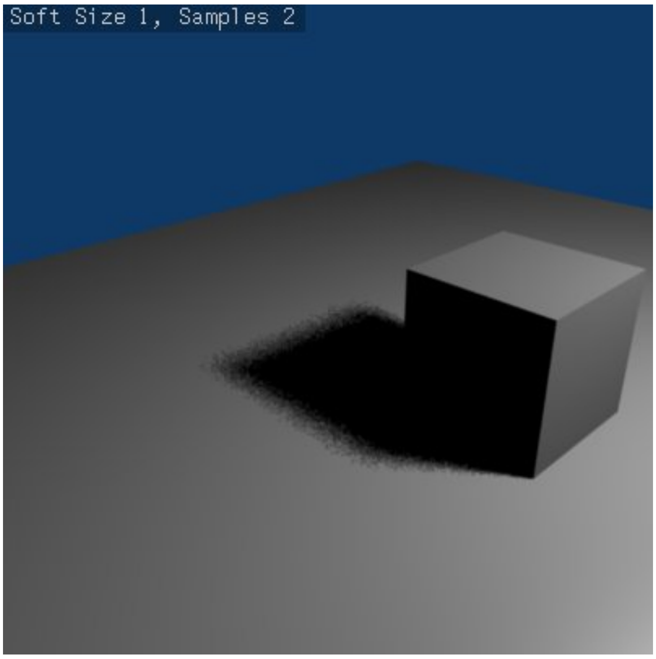




Above is a table of renders with different Soft Size and Sample settings showing the effect of

various values on the softness of shadow edges.

Below is an Animated version of the above table of images showing the effects:



You may need to click on the Image to see the Animation.

◊ Threshold
The Threshold field is used with the Adaptive GMC shadow calculation method. The value in the Threshold field is used to determine if Adaptive GMC shadow sample calculation can skipped based on a threshold of how shadowed an area is already. The maximum Threshold value is 1.

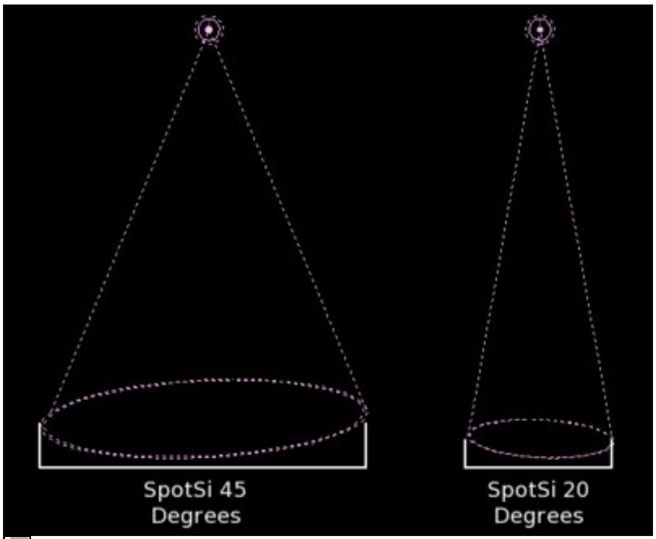
- Only Shadow

  When the Only Shadow button is selected the light source will not illuminate an object but will generate the shadows that would normally appear.

  This feature is often used to control how and where shadows fall by having a light which illuminates but has no shadow, combined with a second light which doesn't illuminate but has Only Shadow enabled, allowing the user to control shadow placement by moving the Shadow Only light around.

- SpotSi:

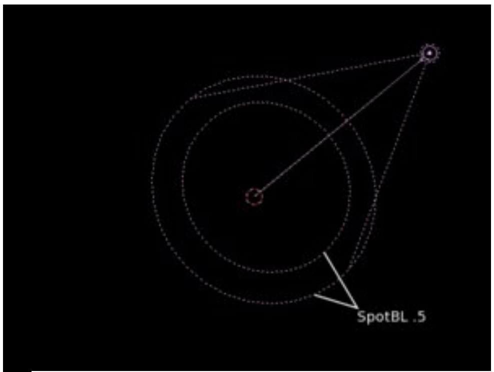
  The SpotSi (SpotSize) Numeric Slider field controls the size of the outer cone of a Spotlight, which largely controls the circular area a Spotlights light covers. It does this by altering the Angle from the position of the Spotlight to the outer cone of the Spotlight. The SpotSi Numeric Slider field represents that Angle. The SpotSi value can be from 1 degree to 180 degrees.



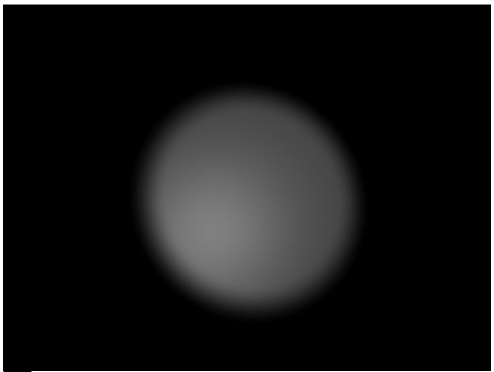
SpotSi settings at 45 and 20 Degrees

- SpotBL:

SpotBL (SpotBLur) Numeric Slider field controls the inner cone of the Spotlight. The SpotBL value can be between 0 and 1. The value is proportional and represents that amount of space that the inner cone should occupy inside the outer cone (SpotSi).



3D View of Spotlight with a SpotBL setting of .5



Render of Spotlight with a SpotBL setting of .5

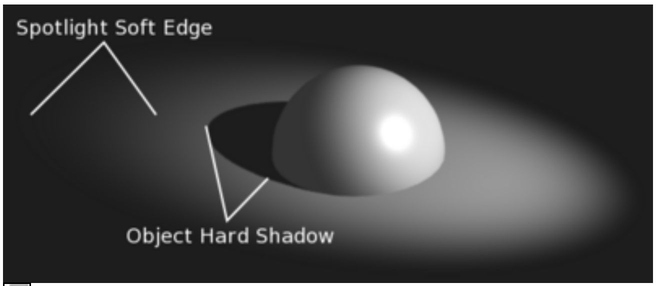
The inner cone boundary line indicates the point at which light from the Spotlight will start to Blur/Soften, before this point the Spotlights light will mostly be full strength. The larger the value of the SpotBL the more Blurred/Soft the edges of the Spotlight will be and the smaller the inner cones circular area will be (as it starts to Blur/Soften earlier).

To make the Spotlight have a sharper falloff rate and therefore less Blurred/Soft edges, decrease the value of SpotBL. Setting SpotBL to 0 results in very sharp Spotlight edges with almost no soft edge.

The falloff rate of the Spotlight light is a ratio between the SpotBL and SpotSi values, the larger the circular gap between the 2 the more gradual the light fades between SpotBL and SpotSi.

You can directly control the effective diameter of the Spotlights circle by adjusting the SpotSi property or indirectly by adjusting the Dist: property. The gap between SpotBL and SpotSi remains constant for changes to Dist:.

SpotBL and SpotSi only control the Spotlight cone's softness or falloff, it does not control the shadow's softness as shown below.

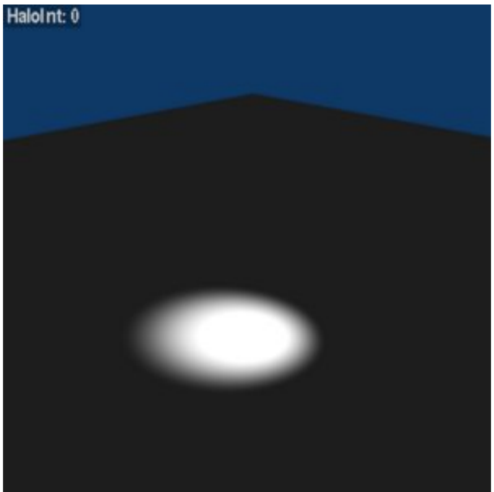


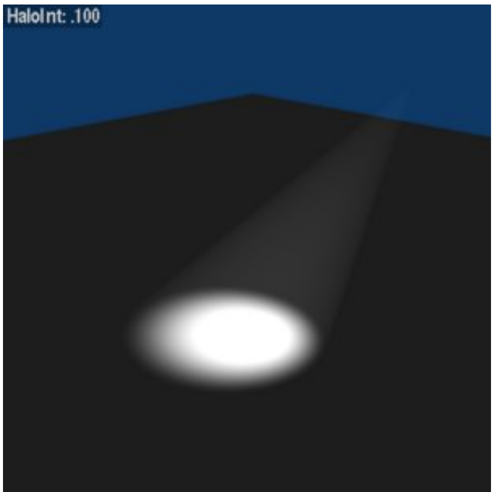
Render showing the Soft Edge Spotlighted area and the Sharp/Hard Object Shadow

Notice in the picture above that the "Object's shadow" is sharp as a result of the raytracing, where as the Spotlights edges are soft. It you want other items to cast soft shadows within the Spotlights area you will need to alter other settings.

- HaloInt:

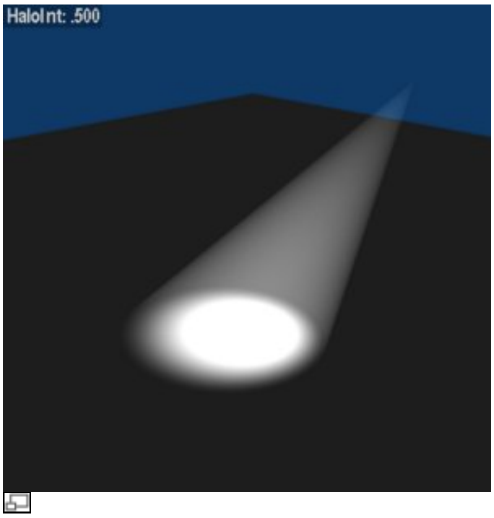
The HaloInt (Halo Intensity) Numeric Slider field controls how Intense/Dense the Volumetric effect is that is generated from the light source. The HaloInt value has a range of between 0 and 5. By Default the value of the HaloInt Numeric Slider field is ignored because the Halo button is not active. To make HaloInt have an effect make sure the Halo button is active. The lower the value of the HaloInt slider the less visible the Volumetric effect is, while higher HaloInt values give a much more noticeable and dense volumetric effect.



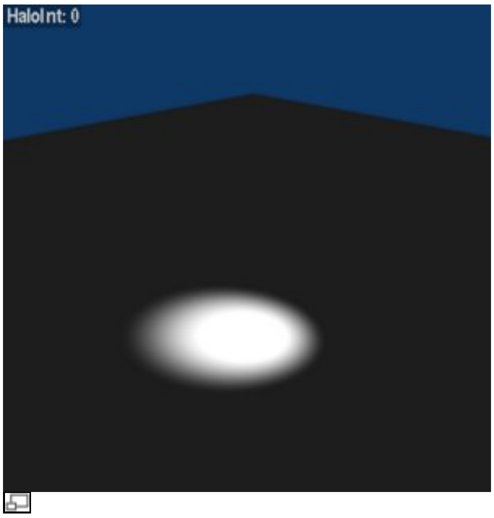


Light with 0 HaloInt

Light with .1 HaloInt



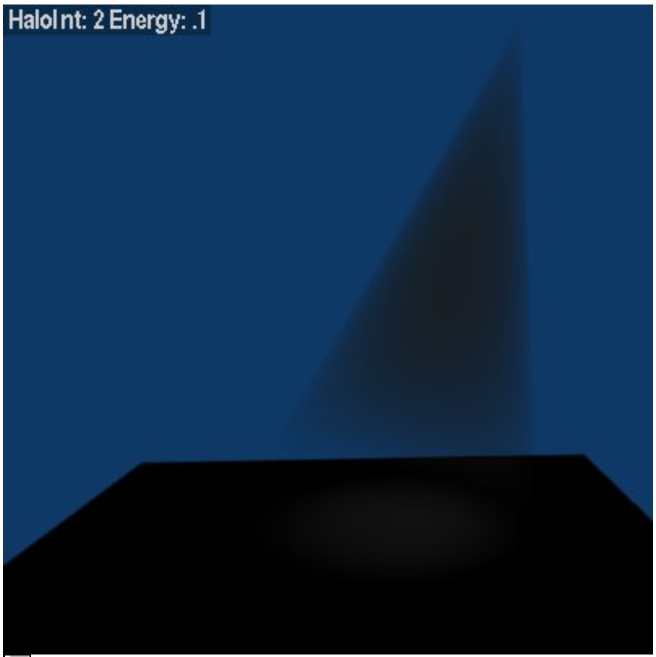
Light with .5 HaloInt



Animation showing the affect of differing HaloInt values, click to see.

Blender only simulates Volumetric lighting in Spotlights when using Blenders Internal Renderer. This can lead to some strange results for certain combinations of settings for Light Energy and HaloInt.

For example having a Spotlight with 0 or very low light Energy settings but a Very high HaloInt setting can result in Dark/Black Halos, which would not happen in the real world. Just be aware of this possibility when using Halos with Blenders Internal Renderer.



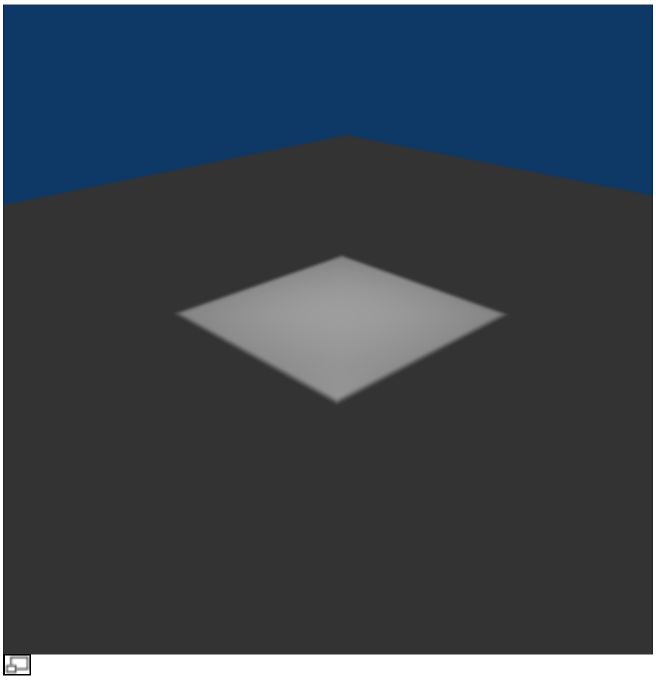
Dark Halo with low Energy lights

- Halo:

  The Halo button allows a Spotlight to have a Volumetric effect applied to it. This button must be active if the Volumetric effect is to be visible.

- Square:

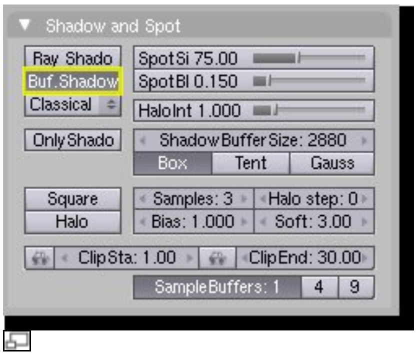
The Square button makes a Spotlight cast a square light area, rather than the default circular one.



Spotlight with a Square light area.

- Buf. Shadow

When the Buf Shadow button is activated, the currently selected Spotlight generates shadows using a Shadow Buffer rather than using Raytracing.



Shadow and Spot Panel Layout with Buf. Shadow button highlighted in yellow.

When the Buf Shadow button is activated, various extra options and buttons appear in the Shadow and Spot panel.

A description of most of these options are listed below:

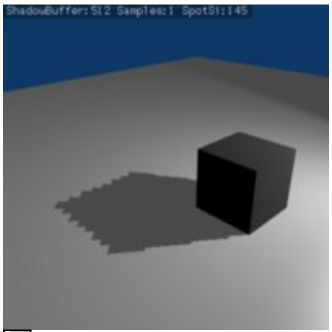
◊ ShadowBufferSize

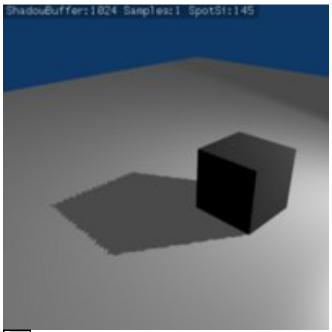
The ShadowBufferSize Numeric Slider field can have a value from 512 to 10240. ShadowBufferSize represents the resolution used to create a Shadow Map. The Shadow Map is then used to determine where shadows lay within a scene.

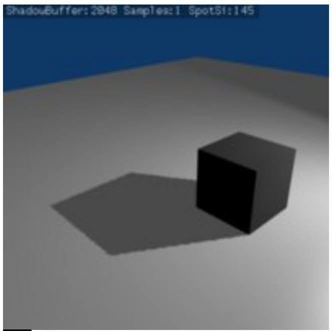
As an example, if you have a ShadowBufferSize with a value of 1024, you are indicating that

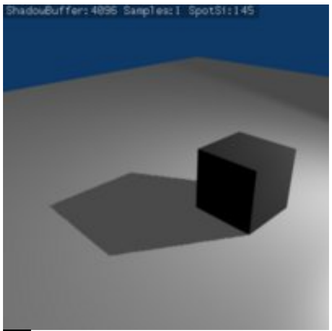
the shadow data will be written to a buffer which will have a square resolution of 1024 pixels/samples by 1024 pixels/samples from the selected Spotlight.

The higher the value of ShadowBufferSize, the higher resolution and accuracy of the resultant shadows, assuming all other properties of the light and Scene are the same, although more memory and processing time would be used. The reverse is also true if the ShadowBufferSize value is lowered, the resultant shadows can be of lower quality, but would use less memory and take less processing time to calculate..





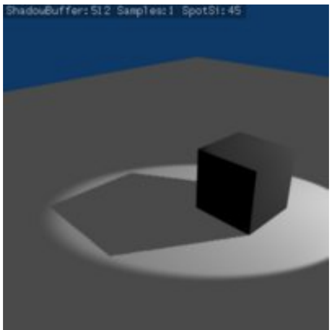


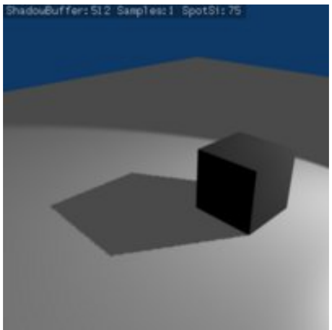


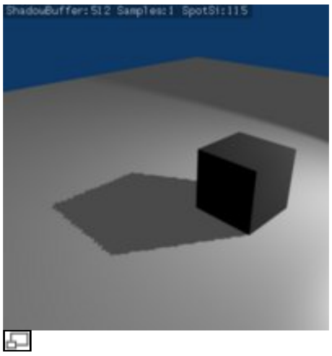
The pictures above show the affect of different ShadowBufferSize values on the quality of shadows. In the above, the filtering has been turned off (Sample value set to 1) from the shadow generation, to make it easier to see the quality degradation of the shadows.

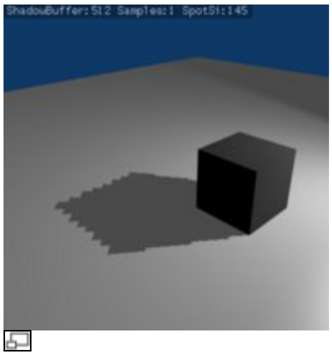
As well as the ShadowBufferSize value affecting the quality of generated shadows, another property of Spotlights that affect the quality of its buffer shadows is the size of the spotlights lighted area (the SpotSi value).

Below are some examples of generated buffer shadows with identical ShadowBufferSize values, but different SpotSi values.









As the SpotSi value is increased it can be seen that the quality of the cast shadows degrade.

This happens because when the Spotlights lighted area is made larger (by increasing SpotSi) the shadow buffer area of the Spotlight would have to be stretched and scaled to fit the size of the new light area. The ShadowBufferSize resolution was not altered to compensate for the change in size of the Spotlight so the quality of the shadows degrade. If you wanted to keep the generated shadows the same quality, as you increased the SpotSi value you would also need to increase the ShaodwBufferSize value.

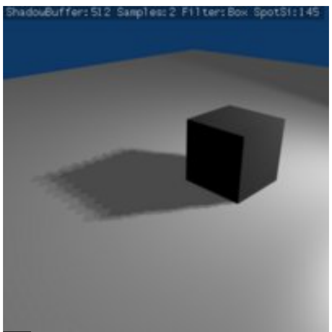
The above basically boils down to:

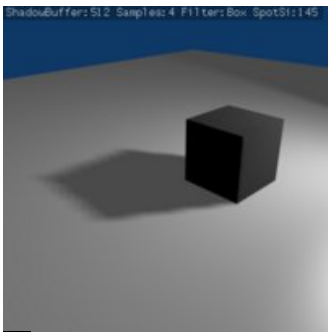
If you have a spotlight that is large you will need to have a larger ShadowBufferSize to keep the shadows good quality. The reverse is true also, if you have a Spotlight which covers a smaller area then the quality of the generated shadows will usually improve (up to a point) as the Spotlight covers a smaller area.

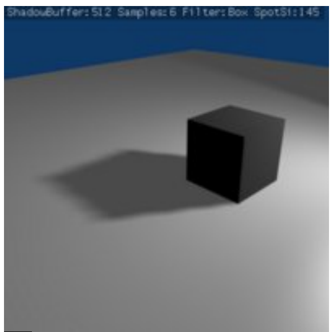
◊ Box

The Box button indicates that shadows generated by buffer ahadow methods will be Anti−Aliased using a Box filtering method.

This is the original filter used in Blender. It is relatively low quality and used for low resolution renders. It produces very sharp Anti−Aliasing. When this filter is used it only takes into account oversampling data which falls within a single pixel and doesn't take into account surrounding pixel samples. It is often useful for images which have sharply angled elements that go up/down/left/right (according to http://arkavision.com/?page_id=125).



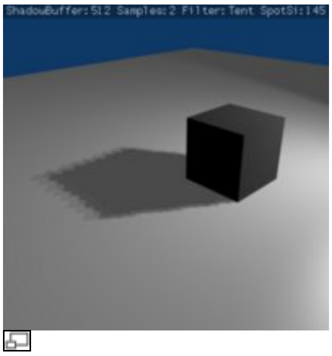


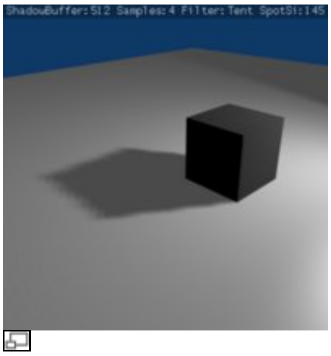


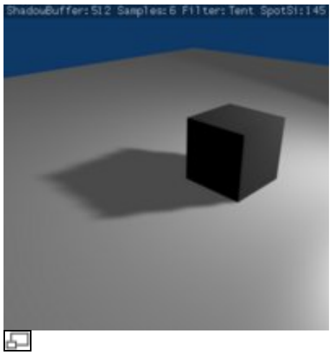
◊ Tent

The Tent button indicates that shadows generated by buffer shadow methods will be Anti−Aliased using a Tent filtering method. It is a simple filter that gives sharp results. It is an excellent general purpose filtering method. This filter also takes into account the sample

values of neighbouring pixels when calculating its final filtering value.

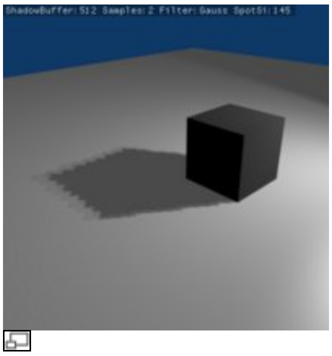


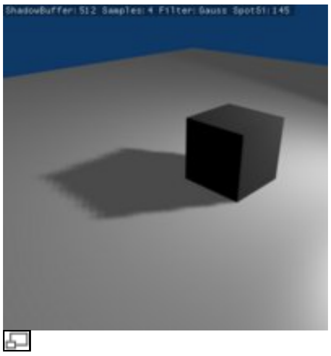


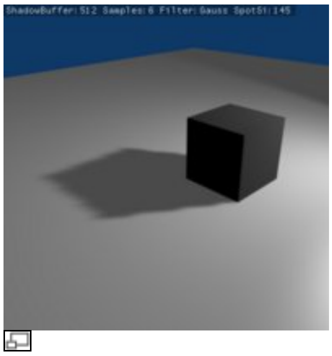


◊ Gauss

The Gauss button indicates that shadows generated by buffer shadow methods will be Anti−Aliased using a Gaussian filtering method. It has a very soft/blurry Anti−Aliasing result. As as result this filter is excellent with high resolution renders.







◊ More Information on Filtering Methods?

The following links will give more information on the various Filtering/Distribution methods and their uses:

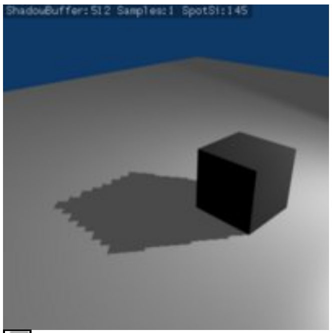
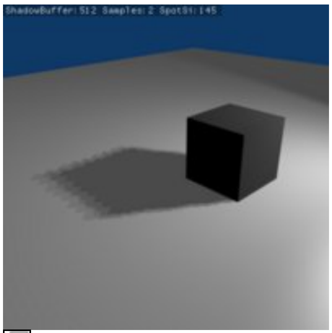
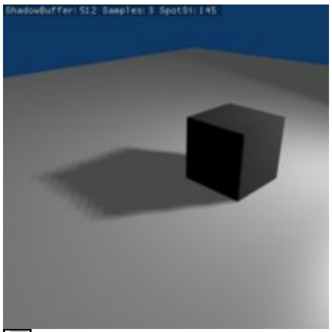
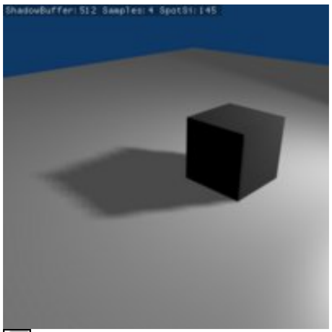
Manual/Oversampling (Antialiasing)

http://arkavision.com/?page_id=125

◊ Samples

The Samples Numeric Slider field can have a value between 1 and 16. It controls the number of samples taken per pixel when calculating shadow maps.

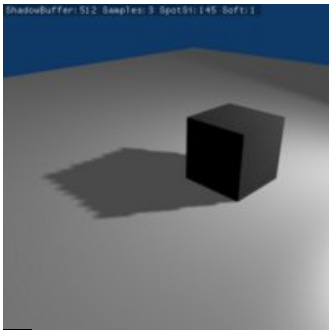
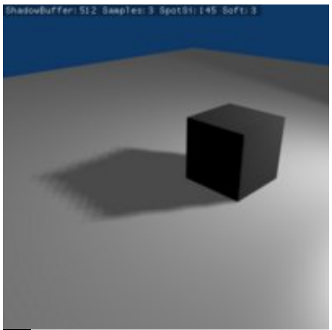
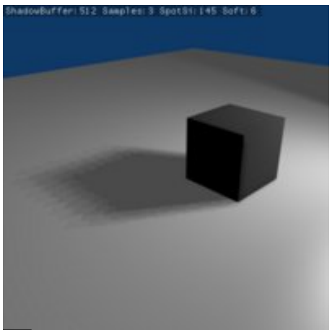
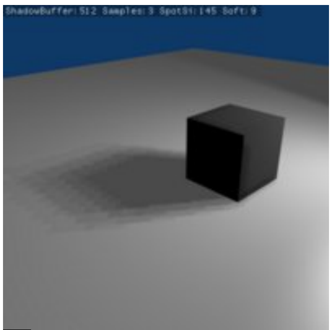
The higher this value the more filtered, smoothed and Anti−Aliased the resultant shadows will be that are generated from the selected light, but the longer they will take to calculate and the more memory will be used. The Anti−Aliasing method used is determined by having one of the Box, Tent or Gauss buttons activated.

As shown above, as the Sample numbers increase the more the edge of shadows become less Aliased/Jaggied. Having a Sample value of 1, is similar to turning off Anti−Aliasing for Buffer Shadows.

◊ Soft

The Soft Numeric Value Slider field can have a value between 1 and 100. The Soft value indicate how wide an area is sampled when doing Anti−Aliasing on buffered shadows. The larger the Soft value the more graduated/soft the area that is Anti−Aliased/softened on the edge of generated shadows.

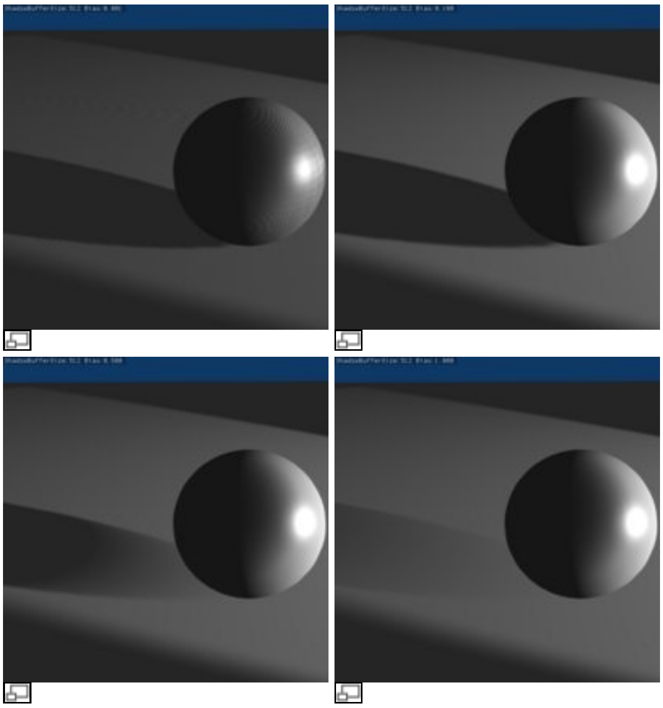
 
 

Above it can be seen that as the Soft size value rises, the softness of the blending of the

shadow edges is more gradated. The blocky Aliasing is exaggerated by altering SpotSi and ShadowBufferSize values in the examples to more easily show the effect of the Soft field.

◊ Bias

The Bias Numeric Slider field can have a value between 0.001 and 5. Bias is used to add a slight offset distance between an Object and the Shadows cast by it. This is sometimes required because of inaccuracies in the calculation which determines weather an area of an Object is in shadow or not. Making the Bias value smaller results in the distance between the Object and its shadow being smaller. If the Bias value is too small an Object can get artefacts which can appear as lines and interference patterns on Objects. When this happens it is usually called "Self Shadowing" and can usually be fixed by increasing the Bias value to prevent self shadowing. Other methods for correcting self shadowing include increasing the size of the ShadowBufferSize or using a different buffer shadow calculation method such as Classic−Halfway or Irregular.



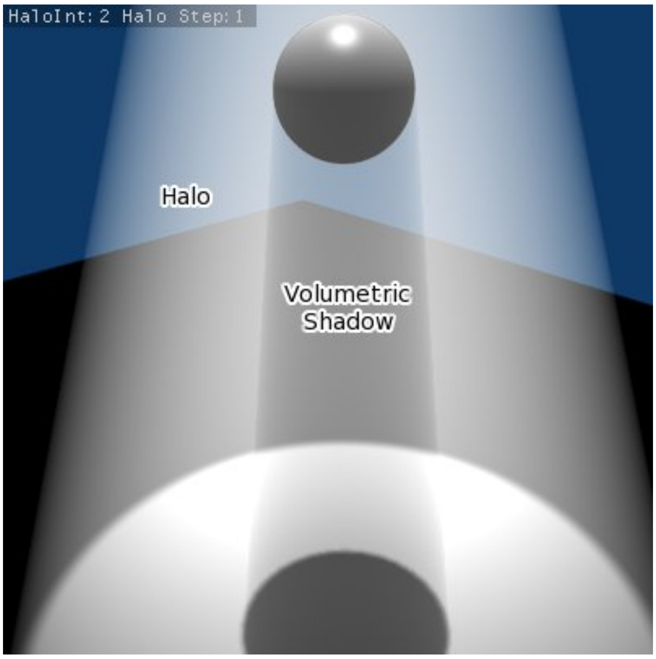
The images above show the affect of different Bias values. With a Bias of 0.001 the scene is full of Self Shadowing interference, but the shadow coming from the back of the sphere is very close to the Sphere. With Bias at 0.100 most of the Self Shadowing interference has been eliminated (apart from small areas on the Sphere), but the start of the shadow point has moved slightly to the left of the Sphere. With Bias values at 0.500 and 1.000 the shadow start point moves even further away from the sphere and there is no Self Shadowing Interference.

Self Shadowing Interference tends to affect curved surfaces more than flat ones, meaning that if your scene has a lot of curved surfaces it may be necessary to increase the Bias value or ShadowBufferSize value.

Having overly large Bias values not only places shadows further away from their casting objects, but can also cause Objects that are very small to not cast any shadows at all. At that point altering Bias, ShadowBufferSize or SpotSi values, among other things may be required to fix the problem.

◊ Halo Step

Halo Step can have a value between 0 and 12. The Halo Step value is used to determine weather a light will cast Volumetric Shadows and what quality those resultant Volumetric Shadows will be.



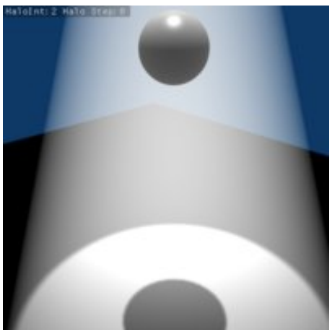
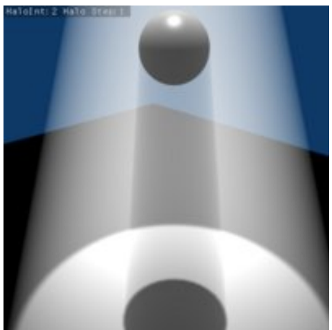
Layout of Halo and Volumetric Shadow

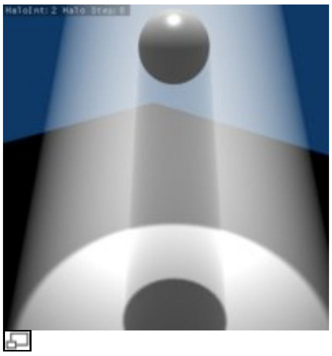
Above is a screenshot which shows a Volumetric Shadow being cast.

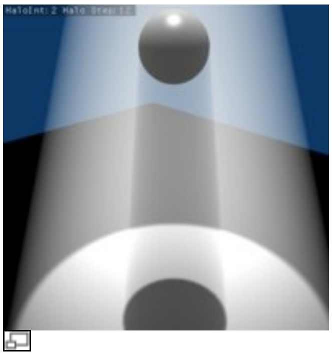
For Volumetric Shadows to work you have to have the Halo button activated and have a high enough HaloInt value, so that the cast Volumetric Shadow is visible. Once these conditions have been met the Halo Step value can be altered to change the quality of Volumetric Shadows.

If Halo Step is set to a value of 0 then no Volumetric Shadow will be generated.

Unlike most other controls, as the Halo Step value increases the quality of Volumetric Shadows decreases (but takes less time to render), whereas when Halo Step value decreases the quality of the Volumetric Shadows increases (but takes more time to render).

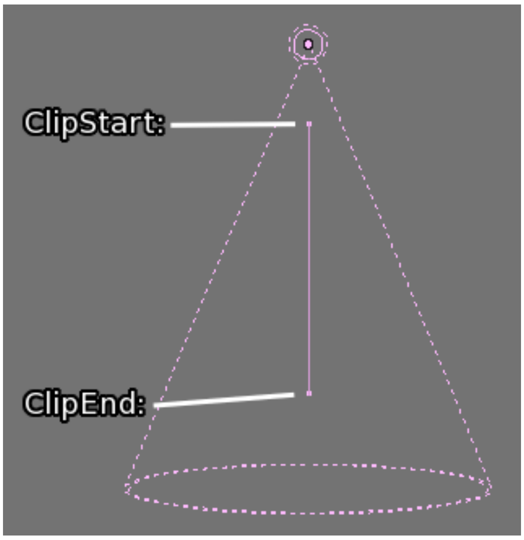







◊ ClipSta & ClipEnd

When a Spotlight with buffered shadows is added to a scene, an extra line appears on the Spotlight, shown below:



Screenshot showing Shadow ClipSta and ClipEnd points line

The start point of the line represents ClipSta;s value and the end of the line represents ClipEnd's value. Both ClipSta and ClipEnd values represent Blender Units.

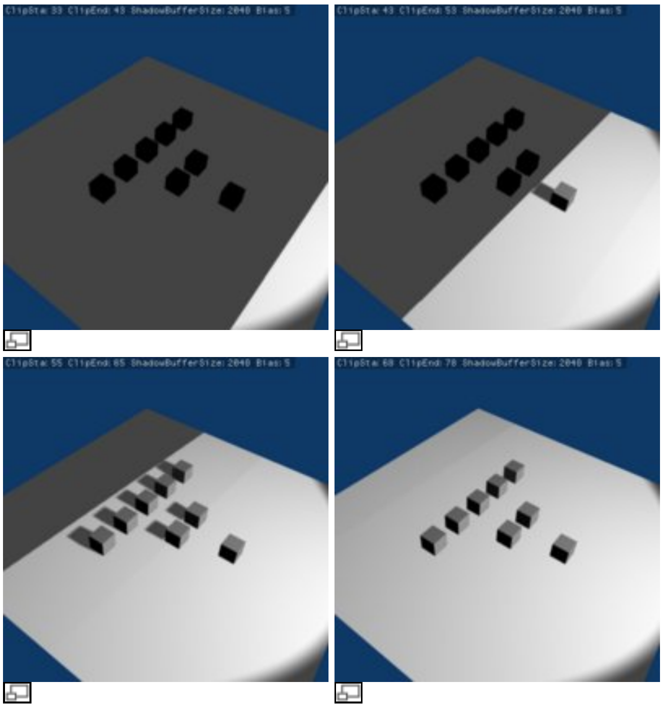
ClipSta can have a value between 0.10 and 1000.

ClipEnd can have a value between 1 and 5000.

Both values are represented in Blender Units.

ClipSta (ClipStart) indicates the point after which Buffered Shadows can be present within the Spotlight area. Any shadow which could be present before this point is ignored and no shadow will be generated.

ClipEnd indicates the point after which Buffered Shadows will not be generated within the Spotlight area. Any shadow which could be present after this point is ignored and no shadow will be generated.



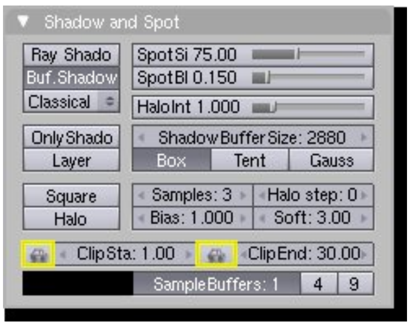
The area between ClipSta and ClipEnd will be capable of having buffered shadows generated.

Altering the ClipSta and ClipEnd values helps in controlling where shadows can be generated. Altering the range between ClipSta and ClipEnd can help speed up rendering, save memory and make the resultant shadows more accurate.

When using a Spotlight with Buffered Shadows, to maintain or increase quality of generated shadows, it is helpful to adjust the ClipSta and ClipEnd such that their values closely bound around the areas which they want to have shadows generated at. Minimising the range between ClipSta and ClipEnd, minimises the area shadows are computed in and therefore helps increase shadow quality in the more restricted area.

◊ Automatic ClipStart & ClipEnd

As well as using the value based ClipSta and ClipEnd fields to control when buffered shadows start and end, it is also possible to have Blender pick the best value independently for each ClipSta and ClipEnd field.

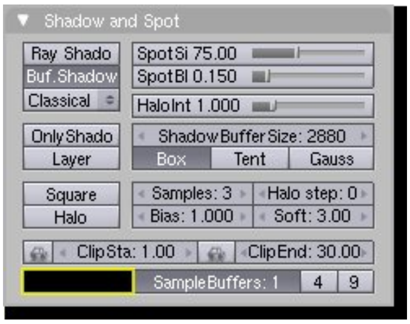


Screenshot showing Automatic ClipSta and ClipEnd buttons, highlighted in yellow.

Blender does this by looking at where the visible vertices are when viewed from the Spotlights position.

◊ Shadow Color

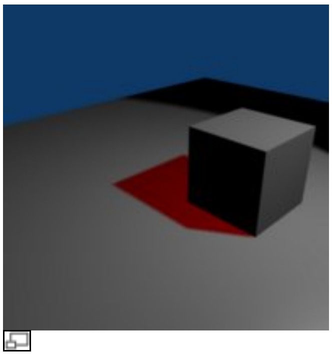
When using buffered shadows it is possible to choose the color of the generated shadow, which does not have to bear any relation to the color of the light lighting the area.

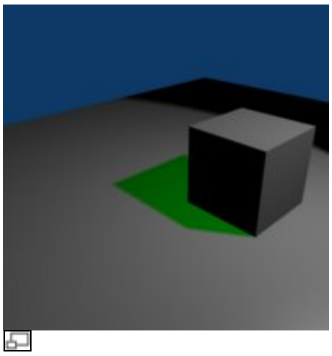


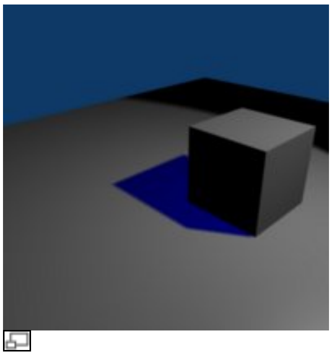
Shadow Color selection box highlighted in yellow.

To change the color from the default (Black), click on the area highlighted in yellow and then select the required color.

This Shadow Color selection box only becomes visible when using Buffered Shadows.


Red Colored Buffer Shadow example


Green Colored Buffer Shadow example

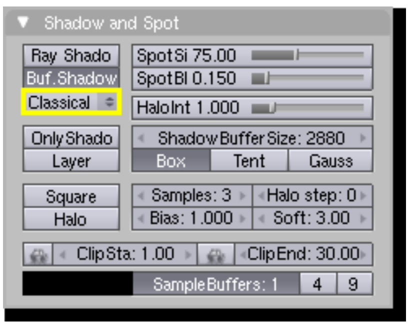

Blue Colored Buffer Shadow example

The above images were all rendered with a white light and the shadow color was selected independently.

Although you can select a pure white color for a shadow color, it appears to make a shadow disappear.

◊ Shadow Buffer Generation Type

Blender has more than one way to generate buffered shadows. The Shadow Buffer Generation Type drop down selector controls which generation type is used for buffered shadow generation. Below the field is highlighted in yellow:



Shadow Buffer Generation Type field highlighted in yellow.

There are 3 shadow generation types, those being:

- Classical
- Classic−Halfway
- Irregular

◊ Classical shadow generation used to be the Blender default method for generation of buffered shadows. It used an older way of generating buffered shadows, but it could have some problems with accuracy of the generated shadows and can be very sensitive to ShadowBufferSize and different Bias values and all the Self−shadowing issues that brings up. It appears that the Classical method of generating shadows is in the position of being obsoleted and is really only still in Blender to works with older versions of Blender, Classic−Halfway should probably be used instead.

◊ Classic−Halfway is an improved shadow buffering method and is currently the default option selected in Blender. It works by taking an averaged reading of the first and second nearest Z depth values allowing the Bias value to be lowered and yet not suffer as much from Self−Shadowing issues. Not having to increase Bias values helps with shadow accuracy, because large Bias values can mean small faces can lose their shadows, as well as preventing shadows being overly offset from the larger Bias value.

Classic−Halfway doesn't work very well when faces overlap, and Biasing problems can happen.

Currently the Halo Step option doesn't work well in some cases. Especially when using planes (not volumes), errors can be introduced.

◊ Irregular, this shadow method is used to generate sharp/hard shadows that are placed as accurately as ray−traced shadows. This method offers very good performance because it can be done as a multi−threaded process.

The method supports transparent shadows by altering the "A Shad" Numeric Slider value:

The Shad A slider highlighted in yellow.

To use Irregular transparent shadows first select the object which will receive the transparent shadow. Then alter the Shad A value in the Material panel. This will only work when Irregular shadow buffer lighting is used.

For more information on the different shadow generation methods see these links:

◊ http://www.blender.org/development/release−logs/blender−243/irregular−shadow−buffer/
◊ http://www.blendernation.com/2006/10/15/blender−gets−irregular−shadow−buffers/
◊ http://www.blender.org/development/release−logs/blender−243/shadow−buffer−halfway−average/

◊ SampleBuffers: 1, 4, 9

The SampleBuffers setting can be set to values 1, 4 or 9 and represents the number of shadow buffers that will be used when doing Anti−Aliasing on shadows generated using shadow buffering.

The higher the number of the SampleBuffers the smoother the Anti−Aliasing, but when using SampleBuffers of 4 you will use 4 times as much memory and with 9, nine times the memory. So Both memory and processing time can be increased, but you get better Anti−Aliasing on small moving objects.

It seems that this option is used in special cases with very small objects, which move and need to generate really small shadows (such as strands). It appears that ormally pixel width shadows don't seem to anti−alias properly and increasing ShadowBufferSize also doesn't seem to help.

Here is a message from Ton Roosendaal about its reason for being from a log message:

Problem:
> Temporal aliasing of shadowbuffers when small details move (like strands).
>
> In this case it doesn't work to simply increase the shadowbuffer size,
> because strands are pixel−sized. Huge shadowbuffers make strand shadows
> almost disappear. So... the shadowbuffer resolution has to be not too high.
>
> Instead of increasing the buffer size, we then create multiple buffers,
> each on different subpixel positions (a bit like "FSA" :).
>
> So! Shadowbuffer sampling then works as follows;
>
> 1) You take multiple samples in the shadowbuffer, on different locations
> inside (or around) the rendered pixel.
> That option was already available as "Samp" button in Lamps
>
> 2) Set amount of sample buffers. It is default 1, but can be 4 or 9.
>
> The results of setting it to '4' or '9' buffers you can see here:
> http://www.blender.org/bf/filters/index3.html
>
> Actually, deep shadowbuffers could solve it probably too! Anyhoo...

Unclear

I am not really clear on how SampleBuffers and Irregular shadows buffers work so if anyone has more info and came make thing clearer either contact me and I will update the page or update it, same goes for anything else you think should be changed on this page. Terrywallwork − 3 Oct 2008

Shadows

Spotlights can use either raytraced shadows or buffer shadows. Either of the two can provide various extra options. Raytraced shadows are generally more accurate, with extra capabilities such as transparent shadows, although they are quite slower to render. Buffered shadows are more complex to set up and involve more faking, but the speed of rendering is a definite advantage.

For more, detailed information see the [Raytraced shadows] or [Buffered shadows] sections.

What is Quasi−Monte Carlo?

The Monte Carlo method is a method of taking a series of samples/readings of values (any kind of values, such as light values, color values, reflective states) in or around an area at random, so as to determine the correct actions to take in certain calculations which usually require multiple sample values to determine overall accuracy, of those calculations. The Monte Carlo methods tries to be as random as possible, this can often cause areas that are being sampled to have large irregular gaps in them (places that are not sampled/read), this in turn can cause problems for certain calculations (such as shadow calculation).

The solution to this was the Quasi−Monte Carlo method.

The Quasi−Monte Carlo method is also random, but tries to make sure that the samples/readings it takes are also better distributed (leaving less irregular gaps in its sample areas) and more evenly spread across an area. This has the advantage of sometimes leading to more accurate calculations based on samples/reading.

What is Volumetric Lighting?

According to Wikipedia, Volumetric Lighting is as described Below:

> "Volumetric lighting is a technique used in 3D computer graphics to add Tyndall−effect lighting to a rendered scene. The term seems to have been introduced from cinematography and is now widely applied to 3D modelling and rendering especially in the field of 3D gaming. It allows the viewer to see beams of light shining through the environment; seeing sunbeams streaming through an open window is an example of volumetric lighting, also known as God rays.
>
> In volumetric lighting, the light cone emitted by a light source is modeled as a transparent object and considered as a container of a "volume": as a result, light has the capability to give the effect of passing through an actual three dimensional medium (such as fog, dust, smoke, or steam) that is inside its volume, just like in the real world."

A classic example is the Search light with a visible Halo/Shaft of light being emitted from it as the search light sweeps around.

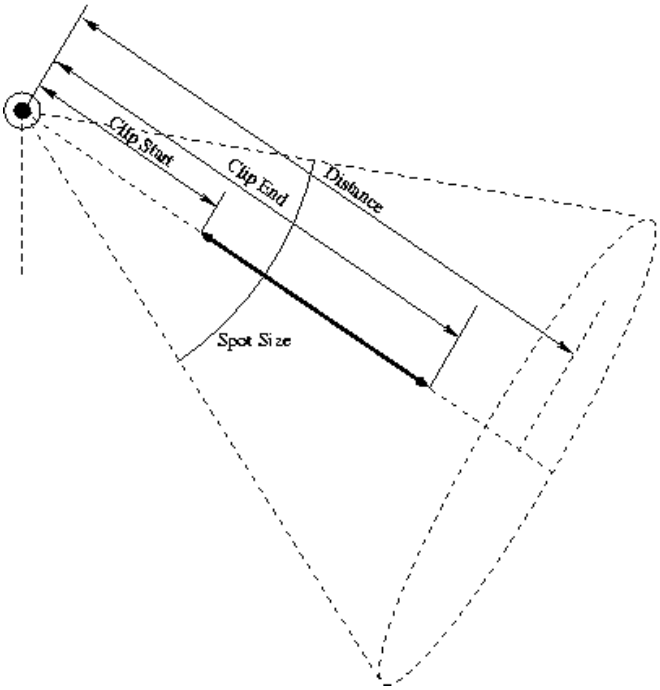
[Blend file of Spotlight Animation]

By default Blender does not model this aspect of light. For example when Blender lights something with a Spotlight you see the Objects and area on the floor lit but not the Shaft/Halo of light coming from the Spotlight as it progresses to its target and would get scattered on the way.

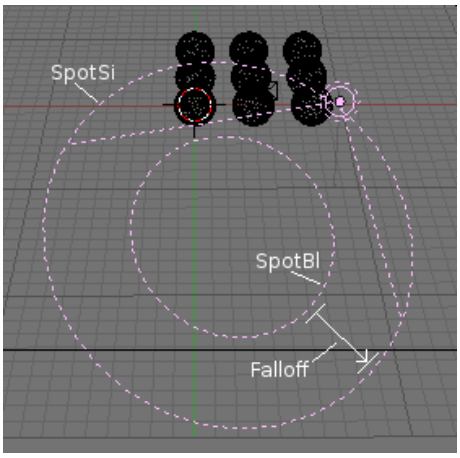
The Halo/Shaft of light is caused in the real world by light being scattered by particles in the air, some of which get diverted into your eye and that you perceive as a Halo/Shaft of light. The scattering of light from a source can be simulated in Blender using various options, but by default is not activated.

Technical Details

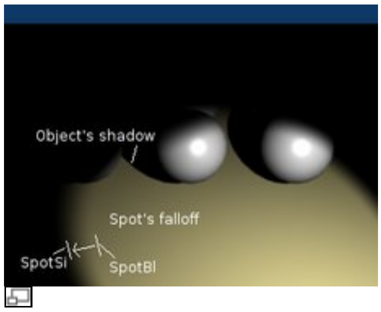
(*Spot Light Scheme*) shows the relationship between the light's properties and how they relate physically.



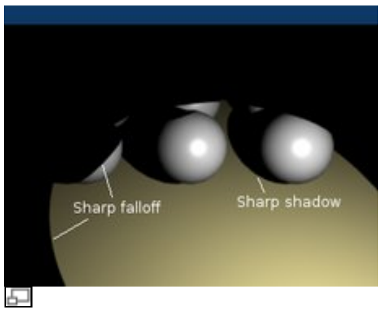
Spot Lamp Scheme



Spot Si and Bl

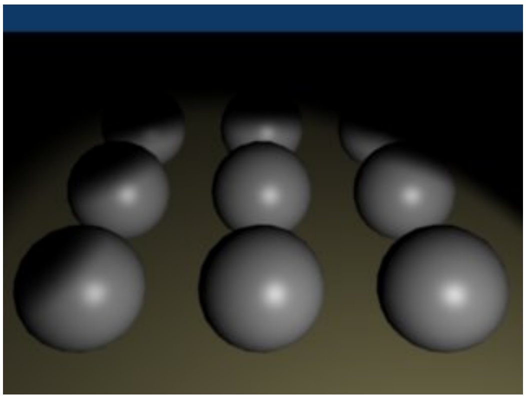


Shadow comparison



Sharp falloff

Examples



Spot lamp render example

See Also

- Raytraced shadows
- Buffered shadows

Previous: Manual/Lamp Contents *Next:* Manual/Area Lamp

User Manual: Contents | Guidelines | Blender Version **2.40**

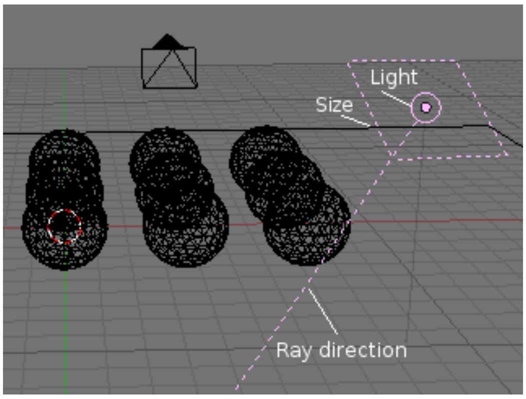
Area Lamp

Mode: All Modes

Panel: Shading/Lamp Context

Hotkey: F5

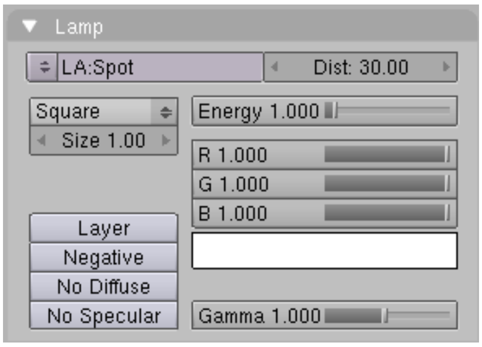
Description



Area Light.

The Area lamp simulates light originating from surface (or surface−like) emitters, for example, a TV screen, your supermarket's neons, a window or a cloudy sky are just a few types. The area lamp produces shadows with soft borders by sampling a lamp along a grid the size of which is defined by the user. This is in direct contrast to point−like artifical lights which product sharp borders.

Options



Area Light's Lamp Panel

Dist
: The area lamp's falloff distance. This is much more sensitive and important for area lamps than for other lamps, usually any objects within the range of Dist will be blown out and overexposed. For best results, set the Dist to just below the distance to the object that you want to illuminate.

Gamma

Amount to gamma correct the brightness of illumination. Higher values give more contrast and shorter falloff.

The Area light replaces the Quad and Sphere buttons with Shape and Size controls. The first one lets you choose the shape of the area and the second the size of the shape.

Square

Emit light from a square area

Size

The width of the square's edge

Rect

Emit light from a rectangular area

SizeX

The rectangle's horizontal width

Size Y

The rectangle's vertical height

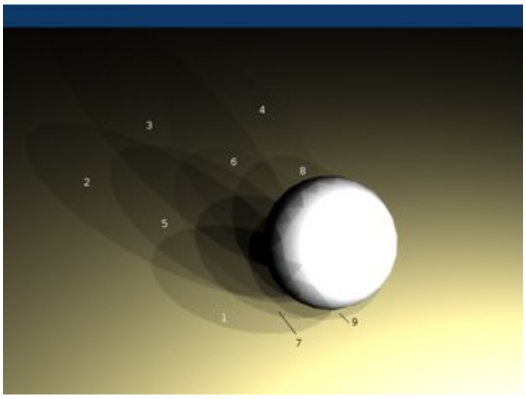
Shape Tips

Choosing the appropriate shape for your Area Light will enhance the believability of your scene. For example, you may have an indoor scene and would like to simulate light entering through a window. You could place a *Rect* area lamp in a window (vertical) or from neons (horizontal) with proper ratios for *SizeX* and *SizeY*. For the simulation of the light emitted by a TV−screen a vertical Square area lamp would be better in most cases.

Shadows

Area lamps can make soft shadows, using raytracing with a number of samples. For more, detailed information see the [Raytraced shadows] sections.

Examples

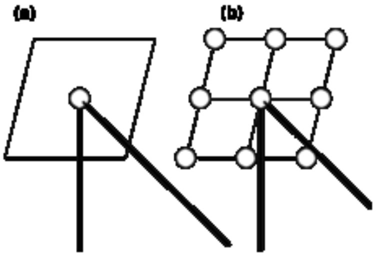


Render example.

In (*Render example*), only one sphere is visible in order to emphasize the shadows created by the Area light. Here the Samples has been set to **3** which will generate 3*3 or 9 shadows. In addition, the Size of the square has been made relatively large (**30**) in order to exaggerate the shadows displaced from one another; the numbers are marked in an arbitrary order. Think of the Size as pushing the lights away from each other in the

plane of the square.

Technical Details



Principles behind the Area Light

The following picture (*Principles behind the Area Light*) helps to understand how the soft shadows are simulated.

(a) is the Area Light as defined in Blender. If its shape is Square, then the softness of the shadow is defined by the number of light Samples in each direction of the shape. For example, (b) illustrates the equivalent case of an Area Light (Square shape), with Samples set at **3** on the Shadow and Spot panel; see Area Light Buttons section.

The Area Light is then considered as a grid with a resolution of 3 in each direction, and with a Light dupliverted at each node for a total of 9 Lights.

In case (a) the energy is "Energy (E)"/**1** and in case (b) the Energy of each individual equivalent Light is equal to E/(No of lights). Each Light produces a faint shadow (proportional to the Energy of the Light), and the overlay of the shadows produces the soft shadows (they are darker where the individual shadows overlap, and lighter everywhere else).

Hints

You will note that changing the Size parameter of your area lamp doesn't affect the lighting intensity of your scene. On the other hand, rescaling the lamp using the S in the 3D View could dramatically increase or decrease the lighting intensity of the scene. This behavior has been coded this way so that you can fine tune all your light settings and then decide to scale up (or down) the whole scene without suffering from a drastic change in the lighting intensity. If you only want to change the dimensions of your Area lamp, without messing with its lighting intensity, you are strongly encouraged to use the Size buttons instead.

With equal Energy and Dist values, an area lamp and a regular lamp will not light the scene with the same intensity. The area lamp will have a tendancy to 'blow out' the highlights, but this can be corrected using the Exp slider in the World buttons.

See Also

- Raytraced shadows

Previous: Manual/Spot Lamp | Contents | *Next:* Manual/Hemi Lamp

User Manual: Contents | Guidelines | Blender Version **2.40**

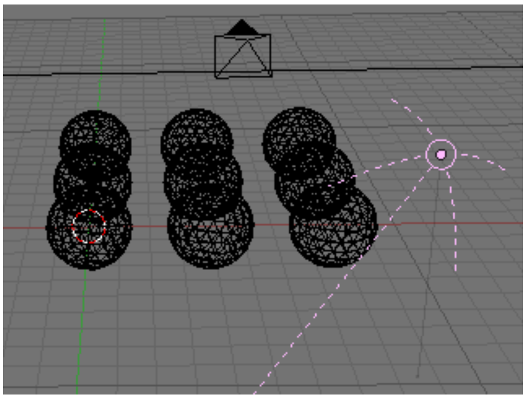
Hemi Lamp

Mode: All Modes

Panel: Shading/Lamp Context

Hotkey: F5

Description



Hemi Light

The Hemi lamp provides light from the direction of a 180° hemisphere, designed to simulate the light coming from a heavily clouded or otherwise uniform sky. In other words it is a light which is shed, uniformly, by a glowing dome surrounding the scene (*Hemi Light conceptual scheme*).

Similar to the Sun lamp, the Hemi's location is unimportant, while its orientation is key.

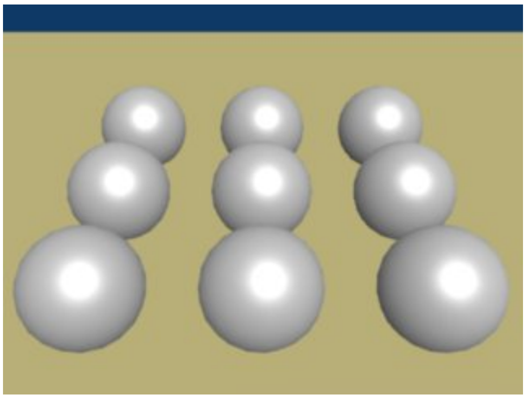
The hemi lamp is represented with four arcs, visualising the orientation of the hemispherical dome, and a dashed line representing the direction in which the maximum energy is radiated, the inside of the hemisphere.

Options

Energy (0.0 − 10.0)
: The intensity of the hemi lamp's illumination

Color
: The color of the hemi lamp's illumination

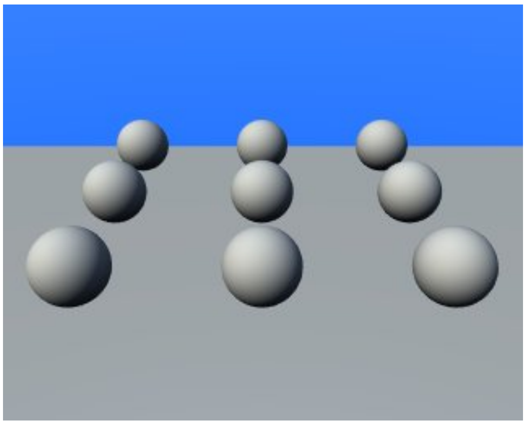
Examples



Render example.

The results of a Hemi Light for the 9 sphere set up are shown in (*Render example*). Notice how the spheres are lit more completely around and to the backside, and the beige ground appears to be bright. The softness of the Hemi light in comparison to the Sun light is evident.

Outdoor Light

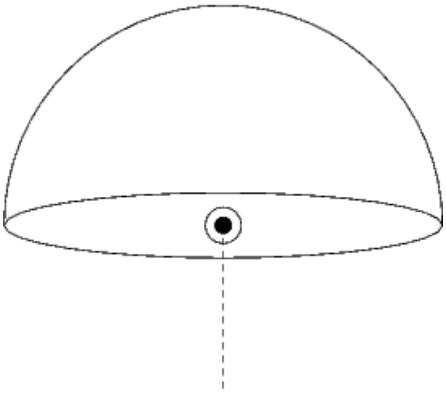


Outdoor Light example

To achieve outdoor lighting you can use both a Sun light, say an Energy of **1.0**, with a warm yellow/orange tint and shadowing enabled, plus a weaker bluish Hemi light faking the light coming from every point of a clear blue sky.

(*Outdoor Light example*) shows an example with relative parameters. The configuration is as: Sun Light Energy equal to **1.0** RGB=(**1.0**,**0.95**,**0.8**) Sun direction in a polar reference is (**135Â°**,**135Â°**). Hemi Light Energy=**0.5** RGB=(**0.64**,**0.78**,**1.0**) pointing down.

Technical Details



Hemi Light conceptual scheme

Previous: Manual/Area Lamp	Contents	*Next:* Manual/Sun Lamp

User Manual: Contents | Guidelines | Blender Version **2.40**

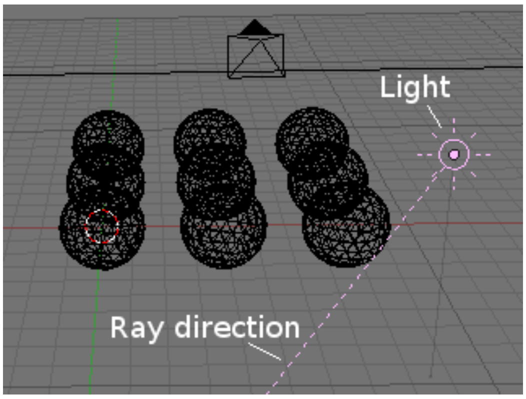
Sun Lamp

Mode: All Modes

Panel: Shading/Lamp Context

Hotkey: F5

Description



Sun Light.

A Sun lamp provides light of constant intensity emitted in a single direction. In the 3D view the Sun light is represented by an encircled black dot with rays emitting from it, plus a dashed line indicating the direction of the light. This direction can be changed by rotating the sun lamp, as any other object, but because the light is emitted in a constant direction, the location of a sun lamp does not affect the rendered result.

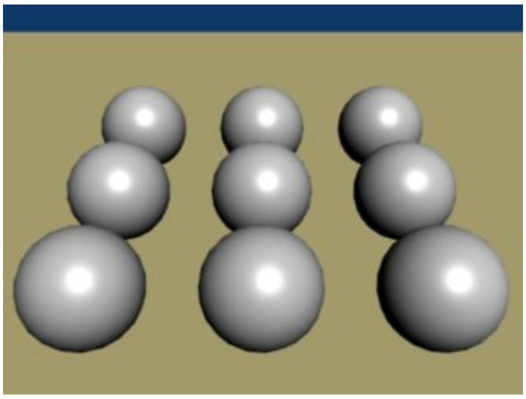
Options

Energy (0.0 − 10.0)
: The intensity of the sun lamp's illumination

Color
: The color of the sun lamp's illumination

Sky/Atmosphere
: Various settings for the appearance of the sun in the sky, and the atmosphere through which is shines, is available. For details click here.

Examples



Render example.

In this rendering the light comes from a constant direction and has a uniform intensity. Notice also, that the white specular highlight on each sphere is in the exact same place; discounting of course the perspective effect from the camera's position being so close to the spheres. This means the actual *location* of the Sun light itself is not important.

Hints

A Sun lamp can be very handy for a uniform clear day−light open−space illumination.

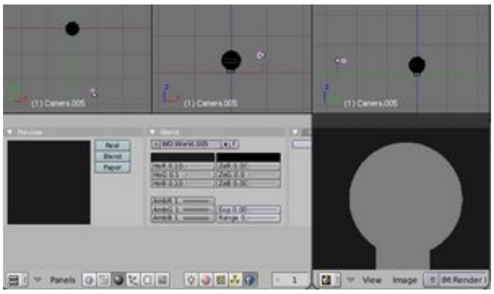
Previous: Manual/Hemi Lamp	Contents	*Next:* Manual/Lighting Rigs

User Manual: Contents | Guidelines | Blender Version **2.43**

A rig is a standard setup and combination of objects; there can be lighting rigs, or armature rigs, etc. A rig provides a basic setup and allows you to start from a known point and go from there. Different rigs are used for different purposes and emulate different conditions; the rig you start with depends on what you want to convey in your scene. Lighting can be very confusing, and the defaults do not give good results. Further, very small changes can have a dramatic effect on the mood and colors. At major studios, lighting is an entire step and specialty. Well, let's get out of the darkness of confusion and let me en*light*en you.

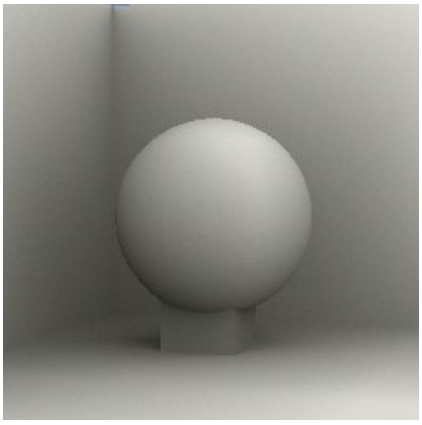
In all the lighting rigs, the default Camera is always positioned 15 degrees off dead−on, about 25 BU back and 9 BU to the side of the subject, at eye level, and uses a long lens (80mm). Up close, a 35mm lens will distort the image. A long lens takes in more of the scene. A dead−on camera angle is too dramatic and frames too wide a scene to take in. So now you know; next time you go to a play, sit off−center and you won't miss the action happening on the sidelines and will have a greater appreciation for the depth of the set. Anyway, enough about camera angles; this is about lighting.

Ambient Only



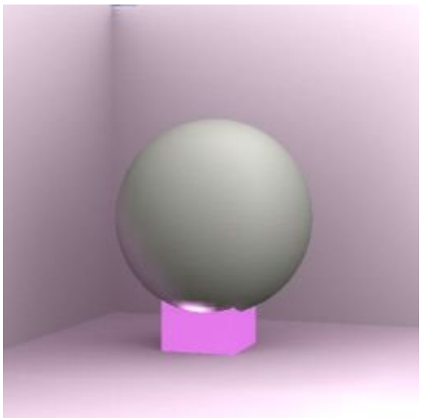
Ambient lighting only

In the Material settings, there is a little globe button where you select the world settings. On the World panel are three sliders AmbR, AmbG, and AmbB, for the color and saturation of ambient light in the world. Ambient light is the scattered light that comes from sunlight being reflected off every surface it hits, hitting your object, and traveling to camera.



Ambient Occlusion

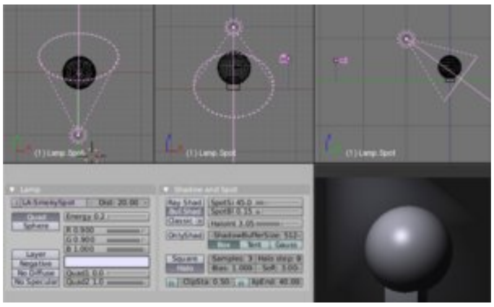
Ambient light illuminates, in a perfectly balanced, shadeless way, without casting shadows. You can vary the intensity of the Ambient light across your scene via [Ambient Occlusion](). In the sample .blend, the "0 pt AO" scene has a light box, Shadow and Ray enabled, and Ambient Occlusion enabled in the World settings. The Ambient color is a sunny white.



Ambient Occlusion with Radiosity

You can add to the Ambient light via [Radiosity](). With Radiosity, the color of light that is radiated from colored objects is mixed with the ambient light. In the sample .blend, the Radiosity scene has Radiosity enabled and the cube is emitting just a little purple to show the effects. You can mix AO and Radiosity as well, as is shown in the example picture to the right and the .blend file in the "0 pt AO+Radio" scene.

Single Rig



Standard spot light rig

The sole, or key, spot light rig provides a dramatic, showy, yet effective illumination of one object or a few objects close together. It is a single spotlight, usually with a hard edge. Halos are enabled in this render to remind you of a smoky nightclub scene. It is placed above and directly in front of the subject; in this case 10 units in front and 10 units high, just like a stage, it shines down at about a 40 degree angle. We use quadractic attenuation, energy of 0.2, a falloff of 20 (the light is about 14 BU from the subject), and a slight coloration of relaxing blue (R:0.9, G:0.9, B:1.0) with a cloud texture mapped to white color at 0.5 mix. This mixing gives some softness to the light. It's a narrow spot at 45 degrees, and the halo you can adjust to your liking.

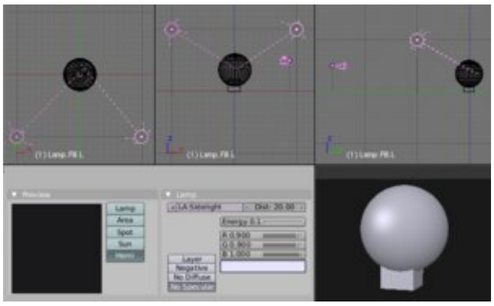
You can make the spot wider by increasing SpotSi and softenting the edge by increasing SpotBl, and parent it to the main actor, so that the spot follows them as they move around. Objects close to the main actor will naturally be more lit and your viewer will pay attention to them.

Moving this spot directly overhead and pointing down gives the interrogation effect. At the opposite end of the show−off emotional spectrum is one soft candelight (short falloff, yellow light) placed really up close to the subject, dramatizing the fearful "lost in the darkness" effect.

Somewhere in the macabre spectrum is a hard spot on the floor shining upward. For fun, grab a flashlight, head into the bathroom and close the door. Turn out the light and hold the flashlight under your chin, pointing up. Look in the mirror and turn it on. Ghoulies! Don't blame me for nightmares, and I hope you get the point: lighting, **even with a single light, varying the intensity, location and direction, changes everything** in a scene.

Use this rig, with Ambient light (and props receiving and being lit by ambient light in their material settings) for scenes that feature one main actor or a product being spotlighted. Do not use this rig for big open spaces or to show all aspects of a model.

Two−Point Rig



Standard 2−point light rig

The two−point lighting rig provides a balanced illumination of an object. Shown to the right are the views of the standard two−point lighting rig. It is called the 2−point because there are two points of light. The standard two−point lighting rig provides a balanced illumination of untextured objects hanging out there in 3D space. This rig is used in real studios for lighting a product, especially a glossy one.

Both lights are almost the same but do different things. Both emulate very wide, soft light by being Hemi with a long falloff distance of 20, which provides even lighting front to back. Both are tinged blue (R:0.9, G:0.9, B:1.0), and have a white cloud texture mixed at 50%. If you use a spot light, you will get a shadow. In real life, these lights bounce light off the inside of a silver umbrella.

The setting for the stage left (on your right) light is shown in the material lamp settings. Notice how we use low intensity to bring out the dimensionality of the sphere; I can't stress that enough. Hard, bright lights actually flatten it and make you squint. Soft lights allow your eye to focus. The left lamp is energy 0.17, because it gives a little more face to the camera (it's on the same side as the camera, so it more directly lights up the face for the camera), and so we disable specular so we don't get that shiny forehead or nose.

The lamp on the left however, lets it be known that it is there by enabling specular; specular flare is that bright spot that is off center above midline on the sphere. It functions also as fill, so we turn down its energy to 0.1.

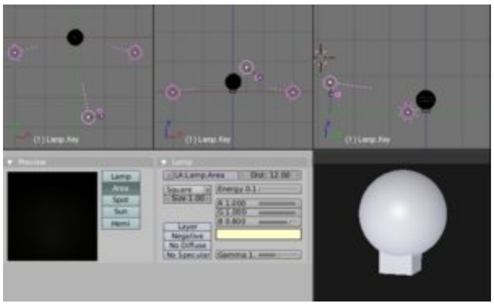
Use this rig to give even illumination of a scene, where there is no main focus. The Hemi's will light up background objects and props, so Ambient is not that important. At the opposite end of the lighting spectrum, two narrow spotlights at higher power with a hard edge gives a "This is the Police, come out with your hands up" kindof look, as if the subject is caught in the crossfire.

Three−Point Rigs

The standard three−point lighting rig is the most common illumination of objects and scenes bar none. If you want to show off your model, use this rig. As you can see, the untextured unmaterialized sphere seems to come out at you. There are multiple thesis on this rig, and you will use one of two:

- Studio − used in a real studio to film in front of a green screen or backdrop. Use this rig when you are rendering your CG objects to alpha into the scene so that the lighting on the actors *and* your CG objects is the same.
- Standard − used in real life to light actors on a set, and gives some backlighting to highlight the sides of actors, making them stand out more and giving them depth.

Studio rig



Studio 3−point light rig

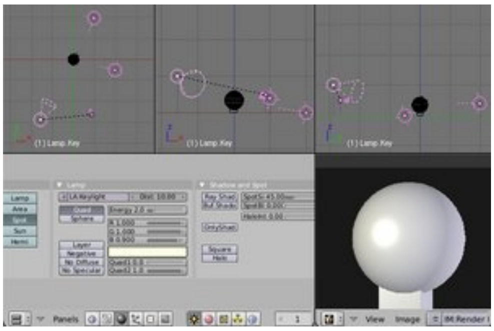
Shown to the right are the Studio top, front, and side views of the standard three−point lighting rig. It changes the dynamics of the scene, by making a brighter "key" light give some highlights to the object, while two side "fill" lights soften the shadows created by the key light. In the studio, use this rig to film a talking head (actor) in front of a green screen, or with multiple people, keeping the key light on the main actor. This rig is also used to light products from all angles, and the side fill lights light up the props.

The key light is the Area light placed slightly above and to the left of the camera. It has an energy of 0.1, a falloff of 12, is slightly yellow (1,1,.8), and it allows the specular to come out. It is about 30 bu back from the subject, and travels with the camera. A little specular shine lets you know there's a light there, and that you're not looking at a ghost. In real life, it is a spot with baffles, or blinders, that limit the area of the light.

The two sidelights are reduced to only fill; each of them are Hemi lights placed 20 BU to the side and 5 BU in front of the subject, at ground level. Each have an energy of 0.2, falloff distance 10, and are slightly blue (R:0.9, G:0.9, B:1.0), although I have seen very blue (R:0.67, G:0.71, B:0.9) used effectively. They don't cause a spotshine on the surface by disabling specular, and at ground level, light under the chin or any horizontal surfaces, countering the shadows caused by the key light. Further, a cloud texture is mapped to white at a 50% mix. This mixing simulates what happens in real life and softens the lighting even further.

Use this rig to give balanced soft lighting that also highlights your main actor or object. It combines the best of both the single rig and the two−point rig, providing balanced illumination and frontal highlights. For a wide scene, you may have to pull the sidelights back to be more positioned like the two−point rig.

Standard Rig



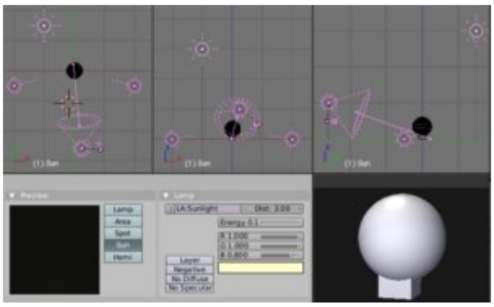
Standard 3−point light rig

Without a curtain in back of your main subject, you have depth to work with. The left fill light has been moved behind the subject (so it is now called a backlight) and is just off−camera, while the right side fill light remains the same. The keylight gives you specular reflection so you can play with specularity and hardness in your object's material settings. The key light gives that "in−the−spotlight" feel, highlighting the subject, while the backlight gives a crisp edge to the subject against the background. This helps them stand out.

In this rig, the key light is a fairly bright spot light with an energy of 2.0 and a falloff of 10. Use a slighter tinge of yellow because the light is so bright; it is the only light for that side. The other sidelight has been moved in back and raised to eye (camera) level. You need to cut the energy of the backlight in half, or when it is added to the remaining sidelight, it will light up the side too much and call too much attention to itself. You can vary the angle and height of the backlight mimic a sun lighting up the objects.

Use this rig in normal 3D animations to light the main actor. Use this rig especially if you have transparent objects (like glass) so that there is plenty of light to shine through them to the camera. The tricky part here is balancing the intensities of the light so that no one light competes with or overpowers the others, while making sure all three work together as a team.

Four−point Rig



4−point light rig

The four−point lighting rig provides a better simulation of outside lighting, by adding a "sun" lamp 30 blender units above, 10 to the side, and 15 BU behind the subject. This sunlight provides backlighting and fills the top of the subject; even producing an intentional glare on the top of their head, telling you there is a sun up there. Notice it is colored yellow, which balances out the blue sidelights.

Changing the key light to a Spot Quad NoSpec, with an energy of 1.0 and pure white light that has a falloff of 12.0 combines with and softens the top sun flare while illuminating the face, resulting in a bright sunshine effect. Two lights above means sharper shadows as well, so you might want to adjust the side fill lights. In this picture, they are still Hemi NoSpec with an energy of 0.1, falloff distance of 10, and blueish speckled as before.

Use this rig when the camera will be filming from behind the characters, looking over their shoulder or whatnot, because the sun provides the backlight there. Also use this rig when you have transparent objects, so there is light to come through the objects to the camera.

Another spot for the fill light is shining up onto the main actor's face, illuminating the underside of his chin and neck. This gets rid of a sometimes ugly shadow under the chin, which if not corrected, can make the actor look fat or like they have a double chin; otherwise distracting. It evens out the lighting of the face.

Troubleshooting and a Helpful Blend file

If you run into a problem with your render, where there are really bright areas, or really dark ones, or strange shadows, or lines on your objects, here's what I suggest you do:

1. First, try killing all materials (select objects and press the X to disassociate the object from that material. Don't worry, as long as you don't reload the file, the material is still out there waiting for you to re−assign it once you've figure out the problem). See if you get those problems with just grayness objects. If you don't have the problem anymore, that should tell you that you've got a materials−interacting−with−light problem. Check the material settings, especially Ambient, Reflection and all those little buttons and sliders on the Shaders panel. You can set some lights to affect only certain materials, so if there's an issue with only a few objects being really bright, start with those.
2. Then start killing lights; regress all the way back to one light, make sure it's smooth, then add them in one by one. As they add together, reduce power in the tested ones so they merge cleanly, or consider not adding it at all, or, especially, reduce the energy of the lamp you just introduced.
3. You can also set lights to only light objects on a layer, so again, if some of the gray spheres have weirdness, check for that as well. Again, you may have done some of this accidentally, so sometimes deleting the light and re−adding it with defaults helps you reset to a known−good situation.
4. Negative lights can be very tricky, and make your model blotchy, so pay special attention to your use of those special lights. Shadow−only lights can throw off the look of the scene as well. Overly textured lights can make your scene have random wierd colors. Don't go too far off a slight tinge of blue or yellow or shades of white, or your material may show Blue in the Material buttons but render Green, and you will be very confused.
5. Look at your environment settings; Horizon, Zenith, and Ambient light.

All of the above lighting rigs were done without raytracing or using Yafray or nodes; those are additional complication layers that I cannot cover here. Sorry.

Hopefully, you will want to download the blend file here. Save it on your hard drive, and when you want to start a new scene, do a File−>Append−>(filename)−>Scene−>(and select the rig you want).

Previous: Manual/Sun Lamp	Contents	*Next:* Manual/Raytraced Shadows

User Manual: Contents | Guidelines | Blender Version **2.40**

Shadows

Shadows are a darkening of a portion of an object because light is being partially or totally blocked from illumninating the object. Blender supports the following kind of shadows:

1. Ray−traced Shadows
2. Buffer shadows
3. Ambient occlusion darkening
4. Radiosity

Ambient Occlusion really isn't a shadow based on light per se, but based on geometry. However, it does mimic an effect where light is prevented from fully and uniformly illuminating an object, so it is mentioned here. Also, it is important to mention Ambient lighting, since increasing Ambient decreases the effect of a shadow. In that same vein, Radiosity casts light from one object onto another, and if Radiosity is high, the shadow will not appear either.

You can use a combination of ray−traced and buffer shadows to achieve different results. Even within Ray Traced shadows, different lamps cast different patterns and intensities of shadow. Depending on how you arrange your lamps, one lamp may wipe out or override the shadow cast by another lamp.

Shadows is one of those trifectas in Blender, where multiple things have to be set up in different areas to get results:

1. . The Lamp has to cast shadows (ability and direction)
2. . Opaque object has to block light on its way (position and layer)
3. . Another object's material has to receive shadows (Shadow and TraShadow enabled)
4. . the Render engine has to calculate Shadows (and Ray, if ray−traced shadows are being used)

For example, the simple Lamp, Area, and Sun light has the ability to cast Ray Shadows, but not buffer shadows. The Spot light can cast both, whereas the Hemi light does not cast any. If a sun lamp is pointing sideways, it will not cast a shadow from a sphere above a plane onto the plane, since the light is not traveling that way.

Just to give you more shadow options (and further confuse the issue), lamps and materials can be set to ONLY cast and receive shadows, and not light the diffuse/specular aspects of the object. Also, Renderlayers can turn on/off the Shadow pass, and their output may or may not contain shadow information.

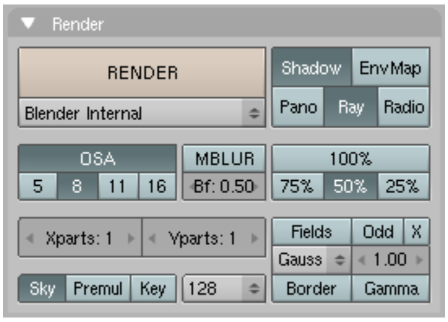
Raytraced shadows

Mode: All Modes

Panel: Shading/Lamp Context Shadow & Spot

Hotkey: F5

Description



Render panel

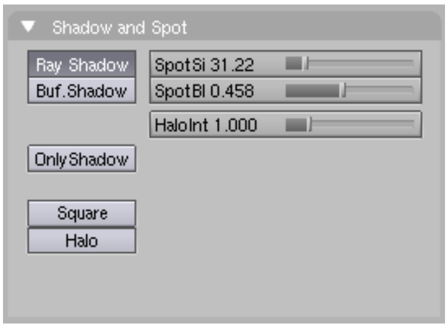
Raytraced shadows produce very precise shadows with very low memory use, but at the cost of processing time. This type of shadowing is available to all lamp types except Hemi.

As opposed to Buffer Shadows, Raytraced shadows are obtained by casting rays from a regular light source, uniformly and in all directions. The raytracer then records which pixel of the final image is hit by a ray light, and which is not. Those that are not are obviously obscured by a shadow.

Each light cast rays in a different way. For example, a Spot Light cast rays uniformly in all directions within a cone. The Sun Light cast rays from a infinitely distant point, with all rays parallel to the direction of the Sun Light.

For each additional light added to the scene, with raytracing enabled, the rendering time increases. Raytraced shadows require more computation than Buffered shadows but produce sharp shadow borders with very little memory resource usage.

Options



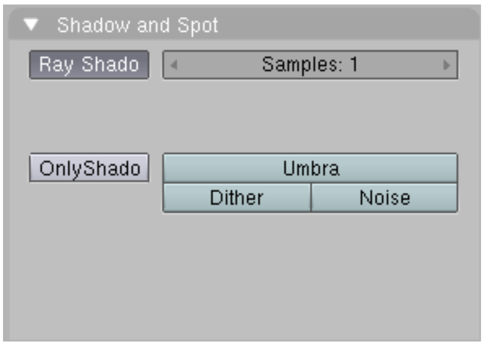
Shadow and Spot panel. Type (Spot)

To enable raytraced shadows two actions are required:

- Enable shadows globally from the Scene context (F10) using the Shadow button on the Render panel; see (*Render panel*).
- Enable Raytracing globally from the same panel using the Ray button.
- Enable shadows for the light using the Ray Shadow button on the Shading context Shadow and Spot panel. This panel varies depending on the type of light.

For further information about the Shadow and Spot panel see the section on Lamp types.

Area lamps



Area Light's Shadow Panel

Area lamps provide additional options for raytraced shadows:

Samples (SamplesX / SamplesY)
: The amount of samples taken to simulate the soft shadow. The more samples, the softer the shadows but the longer it will take to render. For Square area lamps, you have to set only one samples value (Samples). For Rect area lamps, you can set different samples values in the two co−planar directions of the Area lamp (SamplesX and SamplesY).

The following three parameters are intended to artificially boost the "soft" shadow effect, with possible loss in quality:

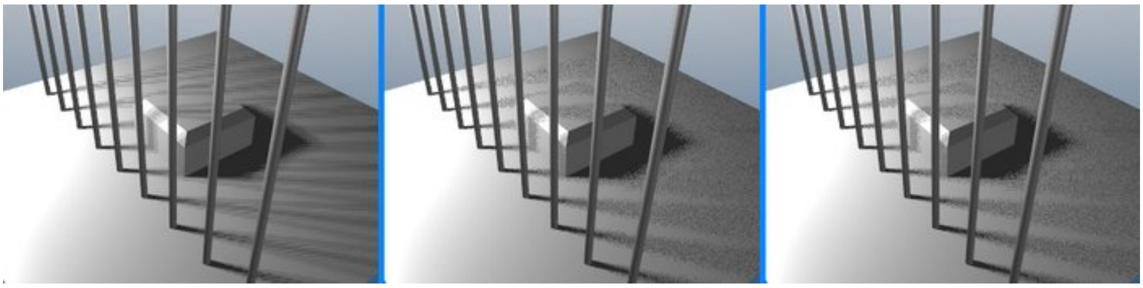
Umbra
: Emphasizes the intensity of shadows in the area fully within the shadow rays. The light transition between fully shadowed areas and fully lit areas changes more quickly (i.e. a sharp shadow gradient). You need Samples values equal to or greater than **2** to see any influence of this button.

Dither
: Applies a sampling over the borders of the shadows, in a similar same way anti−aliasing is applied by the OSA button on the borders of an object. It artificially softens the borders of shadows; when Samples is set very low, you can expect poor results, so Dither is better used with medium Samples values. It is not useful at all with high Samples values, as the borders will appear already soft.

Noise
: Adds noise to break up the edges of solid shadow samples, offsetting them from each other in a pseudo−random way. Once again, this option is not very useful when you use high Samples values where the drawback is that Noise generates quite visible grainyness.

Examples

The first image on the left shows a Samples setting of **2** which generates 4 lights and hence 4 shadows. You can clearly see the shadows, but if you stand back far enough from the image it will appear as a single "soft" shadow. This can be improved by enabling Dithering and improved further by enabling Noise.



Samples 2.0: Dither only, Noise only, and Dither plus Noise

Hints

If your computer isn't very fast, you could find it useful to set a low Samples value (like **2.00**) and activate Dither and/or Noise in order to simulate slightly softer shadows. However, these results will never be better than the same lighting with high Samples values.

Previous: Manual/Lighting Rigs	Contents	*Next:* Manual/Buffer Shadows

User Manual: Contents | Guidelines | Blender Version **2.40**

Buffer Shadows

Mode: All Modes

Panel: Shading/Lamp Context Shadow & Spot

Hotkey: F5

Description

Buffered shadows provides fast rendered shadows at the expense of precision and/or quality. Buffered shadows also require more memory resources as compared to raytracing. Using buffered shadows depends on your requirements. If you are rendering animations or can't wait hours to render a complex scene with soft shadows, buffer shadows are a good choice.

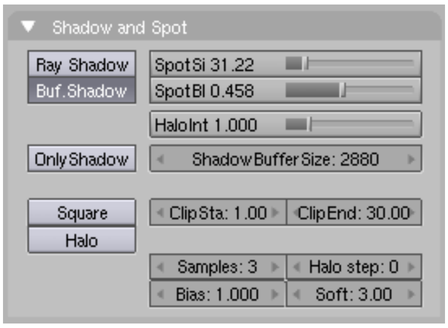
Note

Buffer shadows are only generated from Spot Lights and not from the other types of light source.

Technical Details

For a scanline renderer, the default in Blender, shadows can be computed using a *shadow buffer*. This implies that an *image*, as seen from the spot lamp's point of view is *rendered* and that the distance −−in the image−− for each point from the spot light is saved. Any point in the *rendered* image that is farther away than any of those points in the spot light's image is then considered to be in shadow. The shadow buffer stores this image data.

Options



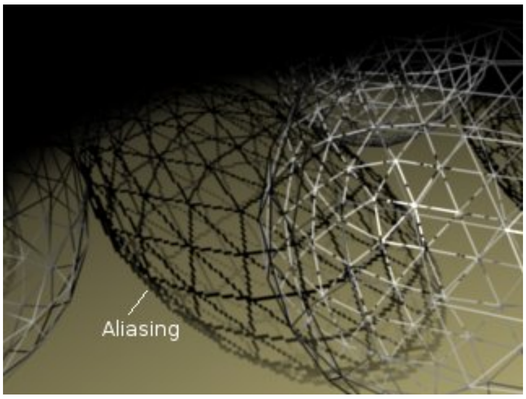
Shadow and Spot panel

BufShadow
: Enable buffer shadows from the active lamp. This reveals additional buttons that control the shadow buffer. Each property can influence the render time and quality of the generated shadows.

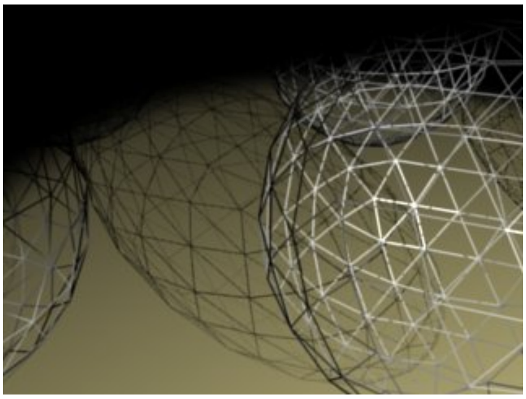
Shadow buffer size

ShadowBuffSize
: The resolution of the shadow buffer (ranging from 512 x 512px to 10240 x 10240px). The higher the value, the more accurate and detailed the shadow will be.

The following image shows a close up view of shadow buffer spotlight casting shadows of an IcoSphere with wire material. Notice the heavily aliased (jagged) shadows of the image (*512 Shadow Buffer size*) compared to the crisp shadows in (*4096 Shadow Buffer size*). Larger shadow buffers look much crisper, but they take longer to generate and use more memory while rendering.



512 Shadow Buffer size.


4096 Shadow Buffer size.

Clipping region

ClipSta/ClipEnd
: The clipping distance within which the shadow buffer is calculated. To further enhance efficiency the shadowis only calculated within a predefined distance from the spot light's position. This range goes from ClipSta, nearer to the spot light, to ClipEnd, further away from the spot light. All objects in between ClipSta and the Spot light are never checked for shadows which results in the objects always being lit. Objects further than ClipEnd are never checked for light and are always in shadow.

To have a realistic shadow ClipSta must be less than the smallest distance between any relevant object of the scene and the spot light, and ClipEnd must be larger than the largest distance.

Note

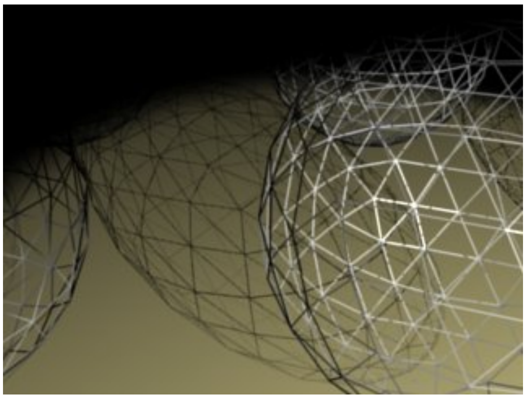
For best shadow quality and most efficient use of resources, make **ClipSta** as large as possible and **ClipEnd** as small as possible, just so it barely encapsulates the objects you want to be shadowed. This minimizes the volume where shadows will be computed and devotes the most resources to the area in focus.

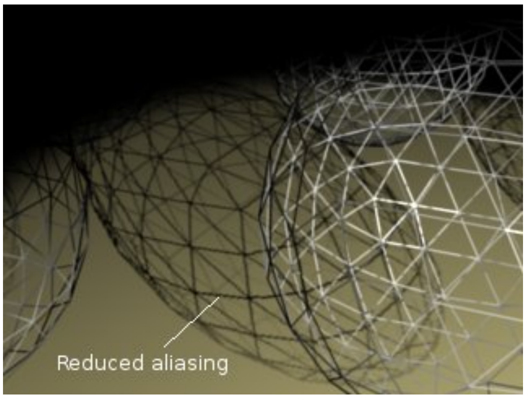
Buffer Sampling

Samples
: The number of samples taken and filtered together for the final shadow result

As well as raising the ShadowBuffSize fore a more accurate result, raising the number of samples can also help by having more data for the renderer to analyse in the shadow calculation. The smaller the ShadowBuffSize the larger Samples: needs to be to have an appreciable effect. Higher Sample values give better anti−aliasing, but require much longer computation times.

Samples work by anti−aliasing the shadow buffer, after it has been computed, by averaging the shadow buffer's pixel value over a square of a side of a given number of pixels. The averaging is performed with a grid applied to each pixel and Samples specifies the size of the grid. A Samples size of 3 means a 3x3 grid.


4096 Shadow Buffer size.



Sample size 3, Shadow buffer size 2048.

In (*Sample size 3, Shadow buffer size 2048*) the aliasing is better than a ShadowBuffSize of 512 but with 1/2 the ShadowBuffSize of 4096. This saves some memory resources but increases the rendering time. You can continue to increase the Samples size but eventually you will reach diminishing returns and increase rendering time.

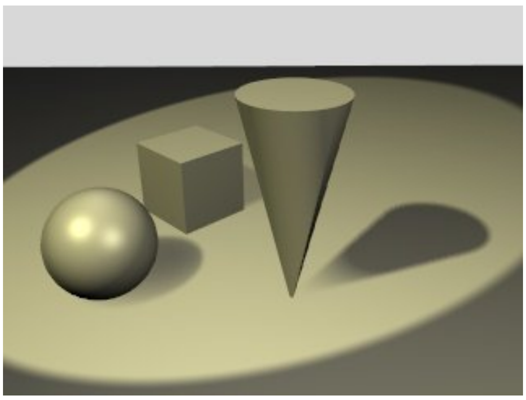
Note

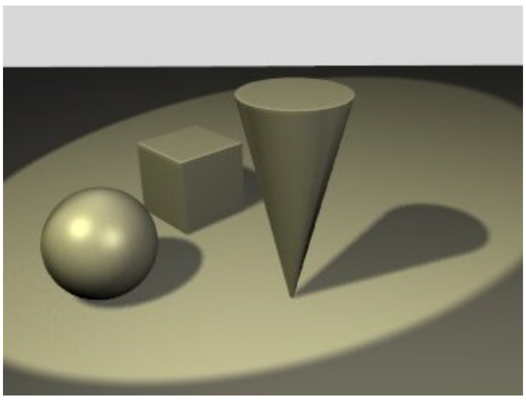
Using a high number of samples is necessary to smooth out soft shadows when using high **Soft** values.

Shadow Bias

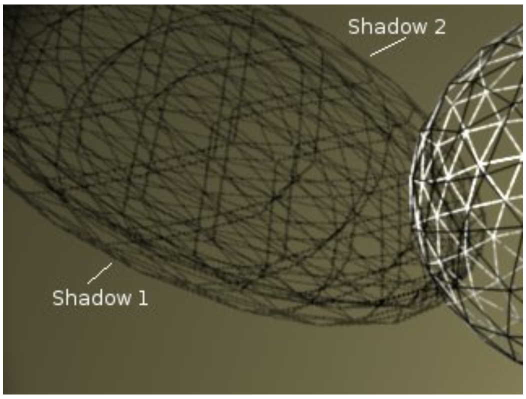
Bias
: Offsets shadows from the object that casts them. Lowering the Bias value moves shadows closer to the object, and can help to fix artifacts at contact points

In the following examples, the default Bias value of 1.0 gives problems in the shadows where the objects come into contact with the floor plane. Lowering the Bias brings the shadows closer to the casting objects to fix this.


Bias: 1.000


Bias: 0.020

Soft Shadows



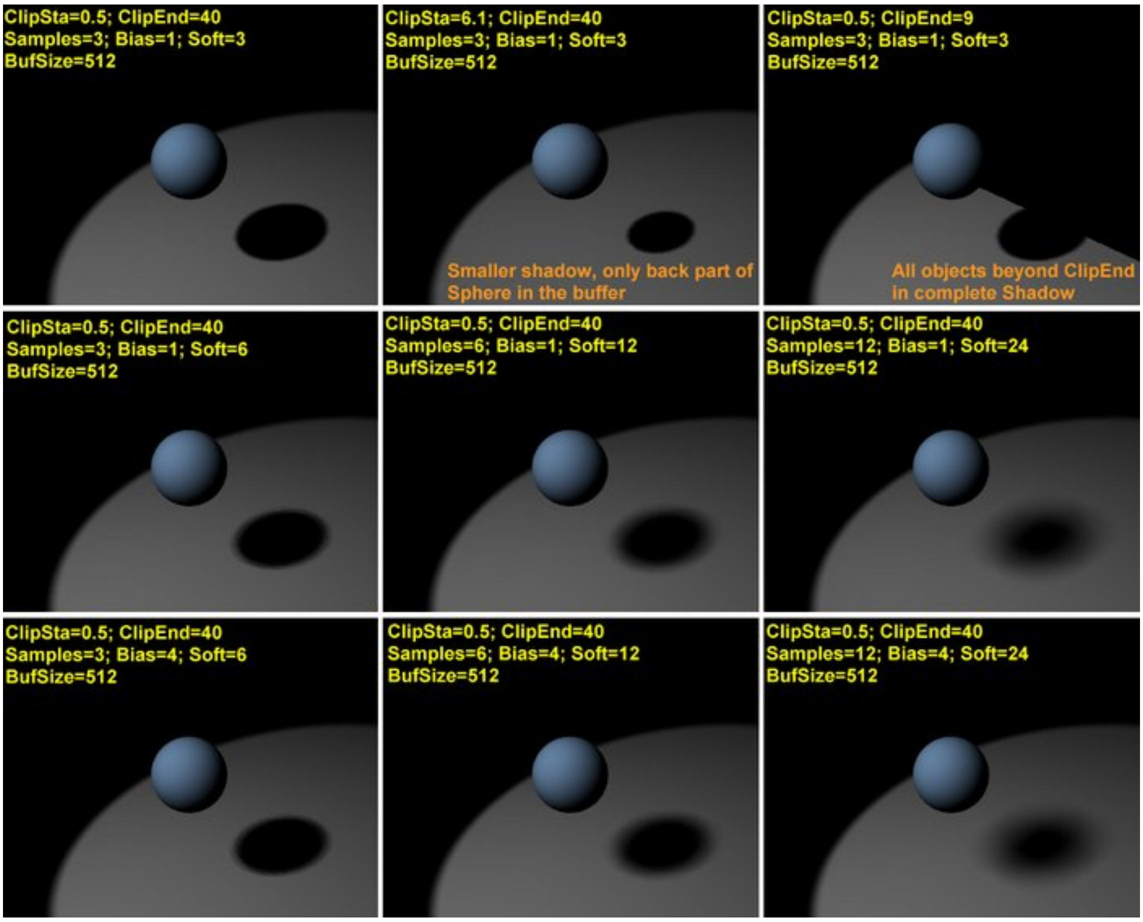
Extreme softness

Soft
: The size of the area over which the buffer samples are scattered and blurred. Softness is similar to the Samples property of the area lamp but instead of multiple lights Soft duplicates shadow samples. You can see this by increasing Soft value far beyond the Samples setting. For smooth results, the Soft's value shouldn't be more than double the Samples value.

(*Extreme softness*) is an example of the Soft value at ten times that of the Samples (**2**) value. Notice the duplicate shadows are obviously very far apart. As with the other buffer properties Soft is an illusionary effect and when taken to extremes the effect becomes obvious.

Examples

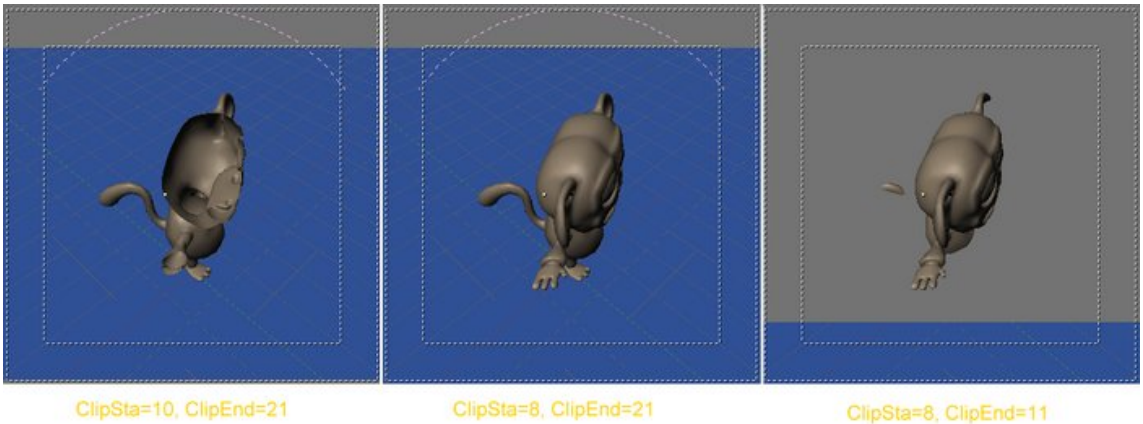
Below are various examples using a solid sphere with shadows cast against a plane object. Note how the different settings affect the shadow that is cast. You get everything from sharp shadows to very soft shadows.



Spot Light shadow examples.

Hints

Any object in Blender can act as a Camera in the 3D view. Hence you can select the Spot Light and switch to a view from its perspective by pressing Ctrl NumPad 0. What you would see, in shaded mode, is shown in (*Spot Light Clipping tweak*)



Spot Light Clipping tweak.

The left frame shows ClipSta too high, the Centre has it correct and the right shows ClipEnd too low. All object(s) nearer to the Spot light than ClipSta and further from ClipEnd is not shown at all. Hence you can fine tune these values by verifying that all shadow *casting* objects are visible.

Previous: Manual/PartV/Ray Shadows	Contents	*Next:* Manual/Volumetric Halos

User Manual: Contents | Guidelines | Blender Version **2.43**

Volumetric Halos

Mode: All Modes (Spot Lamp)

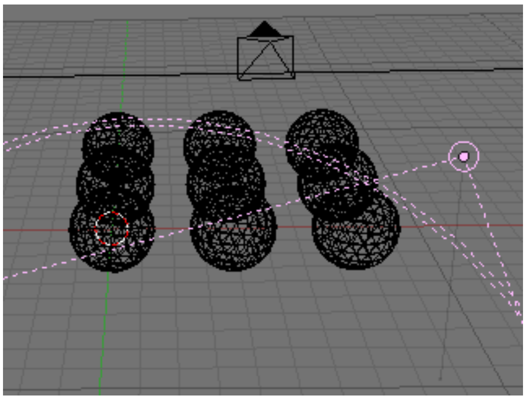
Panel: Shading/Lamp Context Shadow & Spot

Hotkey: F5

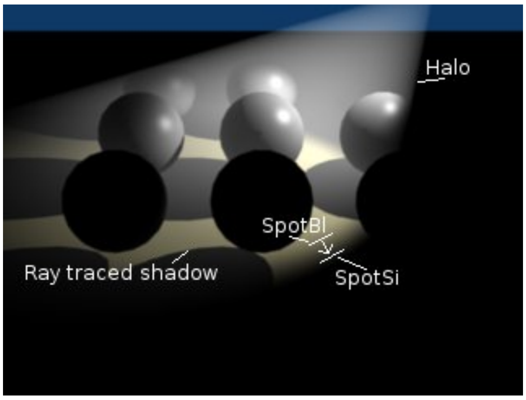
Description

Halo is a volumetric effect, used to simulate light diffusing with an atmosphere (i.e. when light rays become visible as a result of light scattering due to mist, fog, dust etc.). Examples would be smoky bars or foggy environments.

(*Halos*) shows an example of Halo enabled. The light has been moved forward and to the right in order for the cone's halo to be readily visible.



Spot lamp



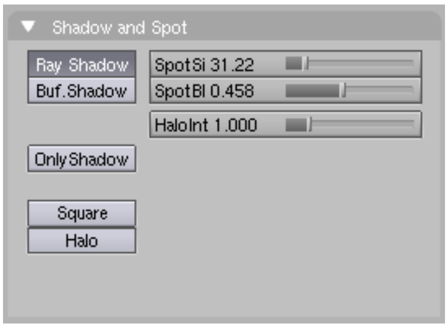
Halos

Options

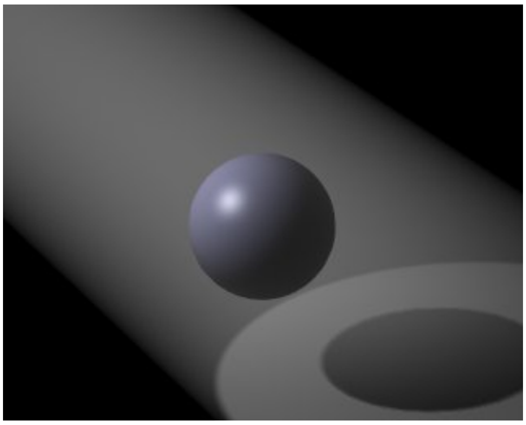
Halos are only available for the Spot lamp.

Halo
: Enable a halo from the active spot lamp

HaloInt
: The intensity of the halo cone, ranging from 0.0 (disabled) to 5.0 (saturated.)



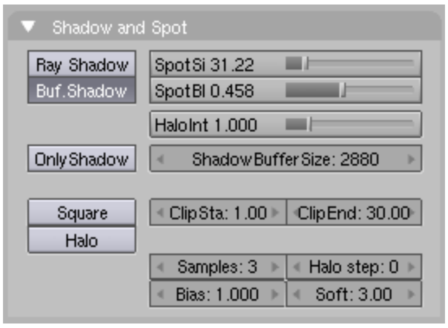
Spot Light Halo button.



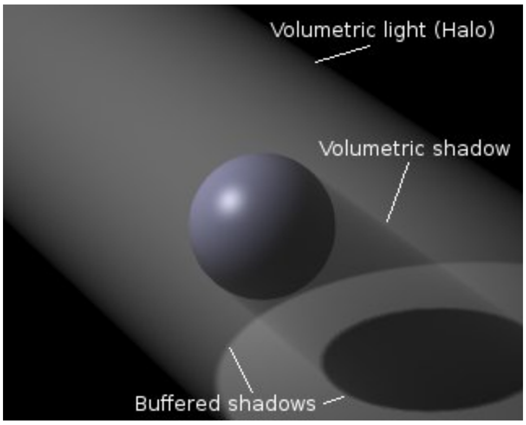
Halo rendering {Raytraced shadows}.

In the above image raytraced shadows are used, which don't support volumetric shadows, so the halo passes right through the sphere. The sphere casts a shadow on the ground, but it should also cast a shadow through the halo as well.

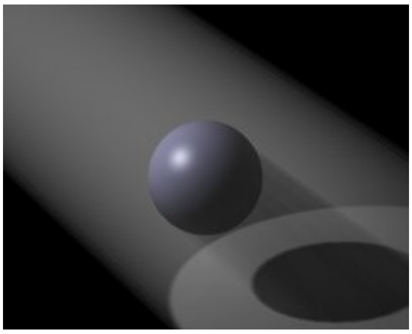
When used with buffer shadows, halos can also cast volumetric shadows.



Spot Light Halo step button.



Halo and shadows rendering {Buffered shadows}.



Halo Step=12

HaloStep
: The number of samples taken in the spot lamp cone. Halo Step's default value of **0** means no sampling at all, which means no volumetric shadow. A value of **1** gives a very fine stepping and better results, but with a slower rendering time (*Halo and shadows rendering {Buffered shadows}*). A higher Halo Step value yields worse results but with faster rendering (*Halo Step=12*).

HaloStep values:

A value of **8** for *Halo Step* is usually a good compromise between speed and accuracy.

Previous: ***Manual/Volumetric Halos***	Contents	*Next:* Manual/Radiosity

User Manual: Contents | Guidelines | Blender Version **2.45**

Introduction

Radiosity example

Most rendering models, including ray−tracing, assume a simplified spatial model, highly optimised for the light that enters our 'eye' in order to draw the image. You can add reflection and shadows to this model to achieve a more realistic result. Still, there's an important aspect missing! When a surface has a reflective light component, it not only shows up in our image, it also shines light at surfaces in its neighbourhood. And vice−versa. In fact, light bounces around in an environment until all light energy is absorbed (or has escaped!). Re−irradiated light carries information about the object which has re−irradiated it, notably colour. Hence not only the shadows are 'less black' because of re−irradiated light, but also they tend to show the colour of the nearest, brightly illuminated, object. A phenomenon often referred to as 'colour leaking' (*Radiosity example*).

In closed environments, light energy is generated by 'emitters' and is accounted for by reflection or absorption of the surfaces in the environment. The rate at which energy leaves a surface is called the 'radiosity' of a surface. Unlike conventional rendering methods, Radiosity methods first calculate all light interactions in an environment in a view−independent way. Then, different views can be rendered in real−time. In Blender, since version 2.28, Radiosity is both a rendering and a modelling tool. This means that you can enable Radiosity within a rendering or rather use Radiosity to paint vertex colours and vertex lights of your meshes, for later use.

Previous: Manual/Volumetric Halos	Contents	*Next:* Manual/Radiosity Rendering

User Manual: Contents | Guidelines | Blender Version **2.45**

Radiosity Rendering

Mode: All Modes

Panel: Render Context Render

Hotkey: F10

Description

Let's assume you have a scene ready, and that you want to render it with the Radiosity Rendering. The first thing to grasp when doing Radiosity is that *no Lamps are necessary*, but some meshes with an Emit material property greater than zero are required, since these will be the light sources. Emit is found on the Shaders panel in the bottom right. Typically, a value of 0.5 or less gives a soft radiance.

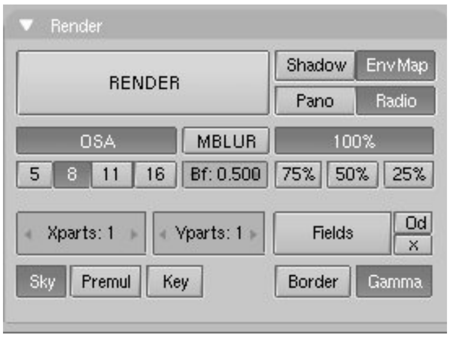
You can build the test scene shown in *Set−up for Radiosity test.*, it is rather easy. Just make a big cube for the room, give different materials to the side walls, add a cube and a stretched cube within it, and add a plane with a non−zero Emit value next to the roof, to simulate the area light.

You assign Materials as usual to the input models. The RGB value of the Material defines the Patch colour. The 'Emit' value of a Material defines if a Patch is loaded with energy at the start of the Radiosity simulation. The 'Emit' value is multiplied with the area of a Patch to calculate the initial amount of unshot energy.

Emitting faces

Check the number of 'emitters' on Blender console! If this is zero nothing interesting can happen. You need at least one emitting patch to have light and hence a solution.

Enabling Radiosity



Enabling Radiosity in the Rendering Buttons.

The *Radio* button on Render panel enables radiosity calculations as part of the render process. This will automatically consider all objects in the scene, and those materials with an Emit: value > 0 will emit light onto other objects

Material Options

When assigning materials, be sure that all of them have Radio enabled on the Links and Pipeline panel.

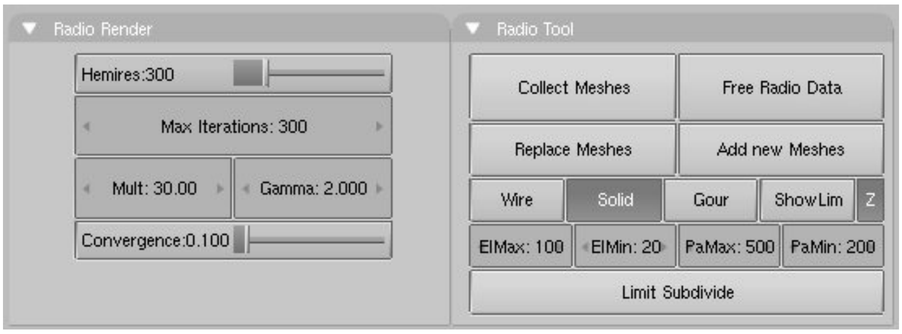
You also need to have the Amb setting of each material > 0 for it to receive emitted light, since the emitted light becomes part of the ambient light. Ambient is found on the Shaders panel.

Objects that emit, or radiate light, should have Emit > 0 as well, also found on the Shaders panel.

Since all objects realistically reflect some light back into the environment, a good general practice is to always set, for every object's material:

- Enable Radio
- Set Ambient > 0, typically 0.1
- Set Emit > 0, typically 0.1

Rendering Options



Radiosity buttons for radiosity rendering.

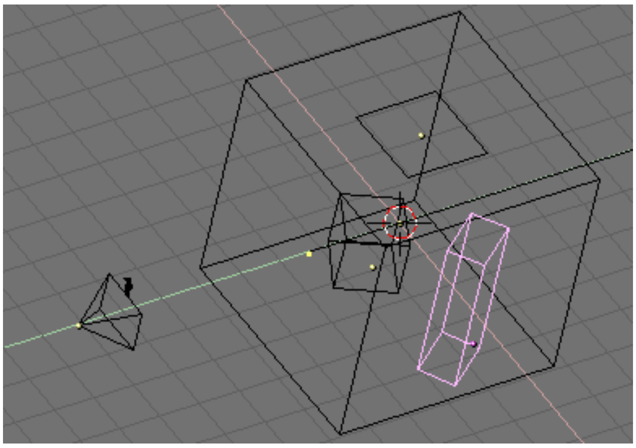
Hemires
: The hemicube resolution; the color−coded images used to find the Elements that are visible from a 'shoot Patch', and thus receive energy. Hemicubes are not stored, but are recalculated each time for every Patch that shoots energy. The 'Hemires' value determines the Radiosity quality and adds significantly to the solving time.

Max Iterations
: The maximum number of Radiosity iterations. If set to zero Radiosity will go on until the convergence criterion is met. You are strongly advised to set this to some non−zero number, usually greater than 100.

Mult, Gamma
: The colourspace of the Radiosity solution is far more detailed than can be expressed with simple 24 bit RGB values. When Elements are converted to faces, their energy values are converted to an RGB colour using the Mult and Gamma values. With the Mult value you can multiply the energy value, with Gamma you can change the contrast of the energy values.

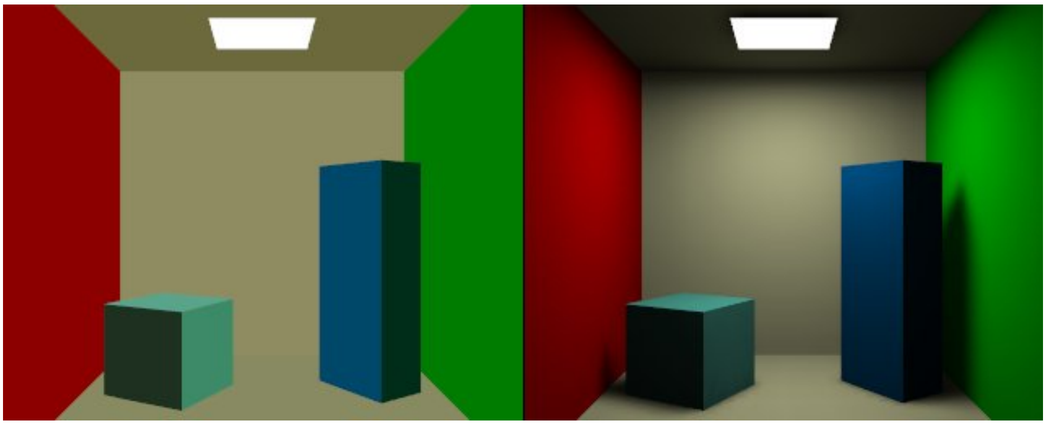
Convergence
: When the amount of unshot energy in an environment is lower than this value, the Radiosity solving stops. The initial unshot energy in an environment is multiplied by the area of the Patches. During each iteration, some of the energy is absorbed, or disappears when the environment is not a closed volume. In Blender's standard coordinate system a typical emitter (as in the example files) has a relatively small area. The convergence value is divided by a factor of 1000 before testing for that reason.

Examples



Set−up for Radiosity test.

The rendering will take more time than usual, in the console you will notice a counter going up. The result will be quite poor (*Radiosity rendering for coarse meshes (left) and fine meshes (right).*, left) because the automatic radiosity render does not do adaptive refinement! Select all meshes, one after the other, and in EditMode subdivide them at least three times. The room, which is much bigger than the other meshes, you can even subdivide four times. Set the Max Iterations a bit higher, 300 or more. Try Rendering again (F12). This time the rendering will take even longer but the results will be much nicer, with soft shadows and colour leakage (*Radiosity rendering for coarse meshes (left) and fine meshes (right).*, right).



Radiosity rendering for coarse meshes (left) and fine meshes (right).

Note

In the Radiosity Rendering Blender acts as for a normal rendering, this means that textures, Curves, Surfaces and even Dupliframed Objects are handled correctly.

Hints

Please note that the light emission is governed by the direction of the normals of a mesh, so the light emitting plane should have a *downward* pointing normal and the *outer* cube (the room) should have the normals pointing inside, (flip them!). Switch to the Radiosity sub−context of the Shading Context. The Panels, shown in *Radiosity buttons for radiosity rendering.*, are two: Radio Rendering which governs Radiosity when used as a rendering tool (present case) and Radio Tool, which governs Radiosity as a modelling tool (next section).

Technical Details

During the late eighties and early nineties Radiosity was a hot topic in 3D computer graphics. Many different methods were developed, the most successful of these solutions were based on the "progressive refinement" method with an "adaptive subdivision" scheme. And this is what Blender uses. To be able to get the most out of the Blender Radiosity method, it is important to understand the following principles:

- *Finite Element Method*

  Many computer graphics or simulation methods assume a simplification of reality with 'finite elements'. For a visually attractive (and even scientifically proven) solution, it is not always necessary to dive into a molecular level of detail. Instead, you can reduce your problem to a finite number of representative and well−described elements. It is a common fact that such systems quickly converge into a stable and reliable solution. The Radiosity method is a typical example of a finite element method inasmuch as every face is considered a 'finite element' and its light emission considered as a whole.

- *Patches and Elements*

  In the Radiosity universe, we distinguish between two types of 3D faces:

  Patches.
  These are triangles or squares which are able to *send energy*. For a fast solution it is important to have as few of these patches as possible. But, to speed things up the energy is modelled as if it were radiated by the Patch's centre; the size of the patches should then be small enough to make this a realistic energy distribution. (For example, when a small object is located above the Patch centre, all energy the Patch sends is obscured by this object, even if the patch is larger! This patch should be subdivided in smaller patches).
  Elements.
  These are the triangles or squares which *receive energy*. Each Element is associated with a Patch. In fact, Patches are subdivided into many small Elements. When an Element receives energy it absorbs part of it (depending on its colour) and passes the remainder to the Patch, for further radiation. Since the Elements are also the faces that we display, it is important to have them as small as possible, to express subtle shadow boundaries and light gradients.

- *Progressive Refinement*

  This method starts with examining all available Patches. The Patch with the most 'unshot' energy is selected to shoot all its energy to the environment. The Elements in the environment receive this energy, and add this to the 'unshot' energy of their associated Patches. Then the process starts again for the Patch *now* having the most unshot energy. This continues for all the Patches until no energy is received anymore, or until the 'unshot' energy has converged below a certain value.

- *The hemicube method*

  The calculation of how much energy each Patch gives to an Element is done through the use of 'hemicubes'. Exactly located at the Patch's center, a hemicube (literally 'half a cube') consist of 5 small images of the environment. For each pixel in these images, a certain visible Element is color−coded, and the transmitted amount of energy can be calculated. Especially with the use of specialized hardware the hemicube method can be accelerated significantly. In Blender, however, hemicube calculations are done "in software". This method is in fact a simplification and optimisation of the 'real' Radiosity formula (form factor differentiation). For this reason the resolution of the hemicube (the number of pixels of its images) is approximate and its careful setting is important to prevent aliasing artefacts.

- *Adaptive subdivision*

  Since the size of the patches and elements in a Mesh defines the quality of the Radiosity solution, automatic subdivision schemes have been developed to define the optimal size of Patches and Elements. Blender has two automatic subdivision methods:

  1. Subdivide−shoot Patches.
  By shooting energy to the environment, and comparing the hemicube values with the actual mathematical 'form factor' value, errors can be detected that indicate a need for further subdivision of the Patch. The results are smaller Patches and a longer solving time, but a higher realism of the solution.
  2. Subdivide−shoot Elements.
  By shooting energy to the environment, and detecting high energy changes (gradients) inside a Patch, the Elements of this Patch are subdivided one extra level. The results are smaller Elements and a longer solving time and maybe more aliasing, but a higher level of detail.

- *Display and Post Processing*

  Subdividing Elements in Blender is 'balanced', that means each Element differs a maximum of '1' subdivide level with its neighbours. This is important for a pleasant and correct display of the Radiosity solution with Gouraud shaded faces. Usually after solving, the solution consists of thousands of small Elements. By filtering these and removing 'doubles', the number of Elements can be reduced significantly without destroying the quality of the Radiosity solution. Blender stores the energy values in 'floating point' values. This makes settings for dramatic lighting situations possible, by changing the standard multiplying and gamma values.

- *Radiosity for Modelling*

  The final step can be replacing the input Meshes with the Radiosity solution (button Replace Meshes). At that moment the vertex colours are converted from a 'floating point' value to a 24 bits RGB value. The old Mesh Objects are deleted and replaced with one or more new Mesh Objects. You can then delete the Radiosity data with Free Data. The new Objects get a default Material that allows immediate rendering. Two settings in a Material are important for working with vertex colours:

  VColPaint.
  This option treats vertex colours as a replacement for the normal RGB value in the Material. You have to add Lamps in order to see the Radiosity colours. In fact, you can use Blender lighting and shadowing as usual, and still have a neat Radiosity 'look' in the rendering.
  VColLight.
  The vertexcolors are added to the light when rendering. Even without Lamps, you can see the result. With this option, the vertex colours are pre−multiplied by the Material RGB colour. This allows fine−tuning of the amount of 'Radiosity light' in the final rendering. As with everything in Blender, Radiosity settings are stored in a datablock. It is attached to a Scene, and each Scene in Blender can have a different Radiosity 'block'. Use this facility to divide complex environments into Scenes with independent Radiosity solvers.

Previous: Manual/Radiosity	Contents	*Next:* Manual/Radiosity Baking

User Manual: Contents | Guidelines | Blender Version **2.45**

Radiosity Tool

Mode: All Modes

Panel: Shading/Radiosity Context

Hotkey: F5

Description

Radiosity can be used also as a tool for defining vertex colours and lights. This can be very useful if you want to make further tweaks to your models, or you want to use them in the Game Engine. Furthermore the Radiosity Modelling allows for Adaptive refinement, whereas the Radiosity Rendering does not! There are few important points to grasp for practical Radiosity Modelling: Only Meshes in Blender are allowed as input for Radiosity Modelling. This because the process generates Vertex colours... and so there must be vertices. It is also important to realize that *each* face in a Mesh becomes a Patch, and thus a potential energy emitter and reflector. Typically, large Patches send and receive more energy than small ones. It is therefore important to have a well−balanced input model with Patches large enough to make a difference! When you add extremely small faces, these will (almost) never receive enough energy to be noticed by the "progressive refinement" method, which only selects Patches with large amounts of unshot energy.

Non−mesh Objects

Only Meshes means that you have to convert Curves and Surfaces to Meshes before starting the Radiosity solution!

Collecting Meshes

Mode: All Modes

Panel: Shading/Radiosity Context

Hotkey: F5

Description

The first step in the process is to convert the selected meshes to radiosity patches, to participate in the solution.

Options

Radio Tool Panel

Collect Meshes
: Convert all selected and visible Meshes in the current Scene to Patches. As a result a new Panel, Calculation, appears. Blender has now entered the Radiosity Modelling mode, and other editing functions are blocked until the

newly created button Free Data has been pressed.

After the Meshes are collected, they are drawn in a pseudo lighting mode that clearly differs from the normal drawing. The Radio Tool Panel (*Gourad button*) has three Radio Buttons:

Wire / Solid / Gour
: These are three drawmode options independent of the indicated drawmode of a 3DWindow. Gouraud display is only performed after the Radiosity process has started. Press the Gour button, to have smoother results on curved surfaces.

Subdivision limits

Mode: All Modes

Panel: Shading/Radiosity Context

Hotkey: F5

Description

Blender offers a few settings to define the minimum and maximum sizes of Patches and Elements.

Options

Radiosity Buttons for Subdivision

Limit Subdivide
: With respect to the values "PaMax" and "PaMin", the Patches are subdivided. This subdivision is also automatically performed when a "GO" action has started.

PaMax, PaMin, ElMax, ElMin
: The maximum and minimum size of a Patch or Element. These limits are used during all Radiosity phases. The unit is expressed in 0.0001 of the boundbox size of the entire environment. Hence, with default 500 and 200 settings maximum and minimum Patch size 0.05 of the entire model (1/20) and 0.02 of the entire model (1/50).

ShowLim, Z
: This option visualizes the Patch and Element limits. By pressing the Z option, the limits are drawn rotated differently. The white lines show the Patch limits, cyan lines show the Element limits.

Adaptive Subdividing

Mode: All Modes

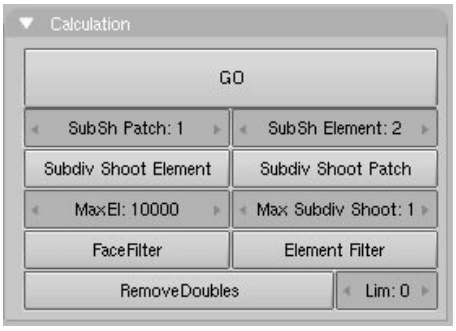
Panel: Shading/Radiosity Context

Hotkey: F5

Description

The last settings before starting the calculations

Options



Radiosity Buttons

MaxEl
: The maximum allowed number of Elements. Since Elements are subdivided automatically in Blender, the amount of used memory and the duration of the solving time can be controlled with this button. As a rule of thumb 20,000 elements take up 10 Mb memory.

Max Subdiv Shoot
: The maximum number of shoot Patches that are evaluated for the "adaptive subdivision" (described below) . If zero, all Patches with 'Emit' value are evaluated.

Subdiv Shoot Patch
: By shooting energy to the environment, errors can be detected that indicate a need for further subdivision of Patches. The subdivision is performed only once each time you call this function. The results are smaller Patches and a longer solving time, but a higher realism of the solution. This option can also be automatically performed when the GO action has started.

Subdiv Shoot Element
: By shooting energy to the environment, and detecting high energy changes (frequencies) inside a Patch, the Elements of this Patch are selected to be subdivided one extra level. The subdivision is performed only once each time you call this function. The results are smaller Elements and a longer solving time and probably more aliasing, but a higher level of detail. This option can also be automatically performed when the GO action has started.

SubSh P
: The number of times the environment is tested to detect Patches that need subdivision.

SubSh E
: The number of times the environment is tested to detect Elements that need subdivision.

Note

Hemires, *Convergence* and *Max iterations* in the *Radio Render* Panel are still active and have the same meaning as in Radiosity Rendering.

GO
: Start the Radiosity simulation. The phases are:

  - ◊ *Limit Subdivide.* When Patches are too large, they are subdivided.
  - ◊ *Subdiv Shoot Patch.* The value of SubSh P defines the number of times the Subdiv Shoot Patch function is called. As a result, Patches are subdivided.
  - ◊ *Subdiv Shoot Elem.* The value of SubSh E defines the number of times the Subdiv Shoot Element function is called. As a result, Elements are subdivided.
  - ◊ *Subdivide Elements.* When Elements are still larger than the minimum size, they are subdivided. Now, the maximum amount of memory is usually allocated.
  - ◊ *Solve.* This is the actual 'progressive refinement' method. The mouse pointer displays the iteration step, the current total of Patches that shot their energy in the environment. This process continues until the unshot energy in the environment is lower than the Convergence value or when the maximum number of iterations has been reached.
  - ◊ *Convert to faces*. The elements are converted to triangles or squares with 'anchored' edges, to make sure a pleasant not−discontinue Gouraud display is possible.

This process can be terminated with Esc during any phase.

Editing the solution

Mode: All Modes

Panel: Shading/Radiosity Context

Hotkey: F5

Description

Once the Radiosity solution has been computed there are still some actions to take

Options

Radiosity post process

Element Filter
: This option filters Elements to remove aliasing artifacts, to smooth shadow boundaries, or to force equalized colours for the RemoveDoubles option.

RemoveDoubles
: When two neighbouring Elements have a displayed colour that differs less than the limit specified in the Lim NumButton, the Elements are joined. The Lim value used here is expressed in a standard 8 bits resolution; a color range from 0 − 255.

FaceFilter
: Elements are converted to faces for display. A FaceFilter forces an extra smoothing in the displayed result, without changing the Element values themselves.

Mult, Gamma
: These NumButtons have the same meaning as in Radiosity Rendering.

Add New Meshes
: The faces of the current displayed Radiosity solution are converted to Mesh Objects with vertex colours. A new Material is added that allows immediate rendering. *The input−Meshes remain unchanged.*

Replace Meshes
: As previous, but the input−Meshes are removed.

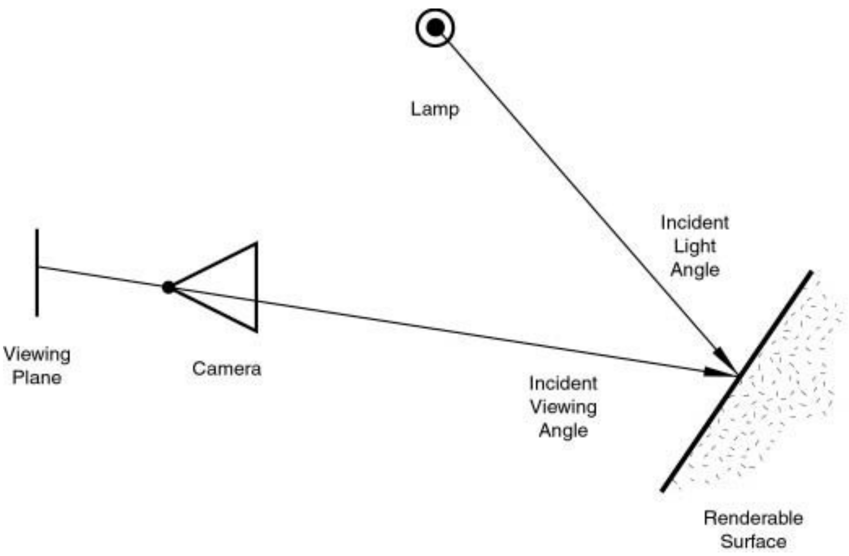
Free Radio Data
: All Patches, Elements and Faces are freed in Memory. You must always perform this action after using Radiosity to be able to return to normal editing.

Previous: Manual/Radiosity Rendering	Contents	*Next:* Manual/Materials

User Manual: Contents | Guidelines | Blender Version **2.45**

Introduction

Before you can understand how to design effectively with materials, you must understand how simulated light and surfaces interact in Blender's rendering engine and how material settings control those interactions. A deep understanding of the engine will help you to get the most from it. The rendered image you create with Blender is a projection of the scene onto an imaginary surface called the *viewing plane*. The viewing plane is analogous to the film in a traditional camera, or the rods and cones in the human eye, except that it receives simulated light, not real light. To render an image of a scene we must first determine what light from the scene is arriving at each point on the viewing plane. The best way to answer this question is to follow a straight line (the simulated light ray) backwards through that point on the viewing plane and the focal point (the location of the camera) until it hits a renderable surface in the scene, at which point we can determine what light would strike that point. The surface properties and incident light angle tell us how much of that light would be reflected back along the incident viewing angle (*Rendering engine basic principle.*).



Rendering engine basic principle.

Two basic types of phenomena take place at any point on a surface when a light ray strikes it: diffusion and specular reflection. Diffusion and specular reflection are distinguished from each other mainly by the relationship between the incident light angle and the reflected light angle. The shading (or coloring) of the object during render will then take into account the base color (as modified by the diffusion and specular reflection phenomenon) and the light intensity

Using the internal raytracer, other (more advanced) phenomena could occur. In raytraced reflections, the point of a surface stroke by a light ray will return the color of its surrounding environment, according to the rate of reflection of the material (mixing the base color and the surrounding environment's) and the viewing angle. On the other hand, in raytraced refractions, the point of a surface stroke by a light ray will return the color of its background environment, according to the rate of transparency (mixing the base color and the background environment's along with its optional filtering value) of the material and the optional index of refraction of the material, which will distort the viewing angle.

Of course, shading of the object hit by a light ray will be about mixing all these phenomena at once during the rendering. The appearance of the object, when rendered, depends on many inter−related settings:

- World (Ambient color, Radiosity, Ambient Occlusion)

- Lights
- Material settings (including ambient, emmission, and every other setting on every panel in that context)
- Texture(s) and how they are mixed
- Material Nodes
- Camera
- viewing angle
- obstructions and transparent occlusions
- shadows from other opaque/transparent objects
- Render settings
- Object dimensions (SS settings are relvant to dimensions)
- Object shape (refractions, fresnel effects)

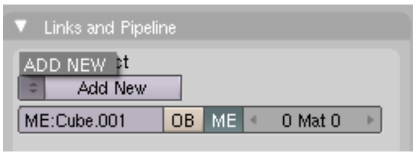
Previous: Manual/Radiosity Baking	Contents	*Next:* Manual/Materials in practice

User Manual: Contents | Guidelines | Blender Version **2.45**

In this section we look at how to set up the various material parameters in Blender, and what you should expect as a result. We also give hints about practical material usage.

Creating a new Material

Every time a new Object is created it has no material linked to it. By pressing the F5 key or clicking on , you switch to the Shading context and the Material Buttons window appears. This window should be almost empty at this point. Adding a new material is done with the menu button shown in *Add new material.*.



Add new material.

Once the object has at least one material linked to it, many new panels (some tabbed) are readily displayed to allow you to precisely control the shading of the material. These settings, together with the Material Nodes, are called a "shader".

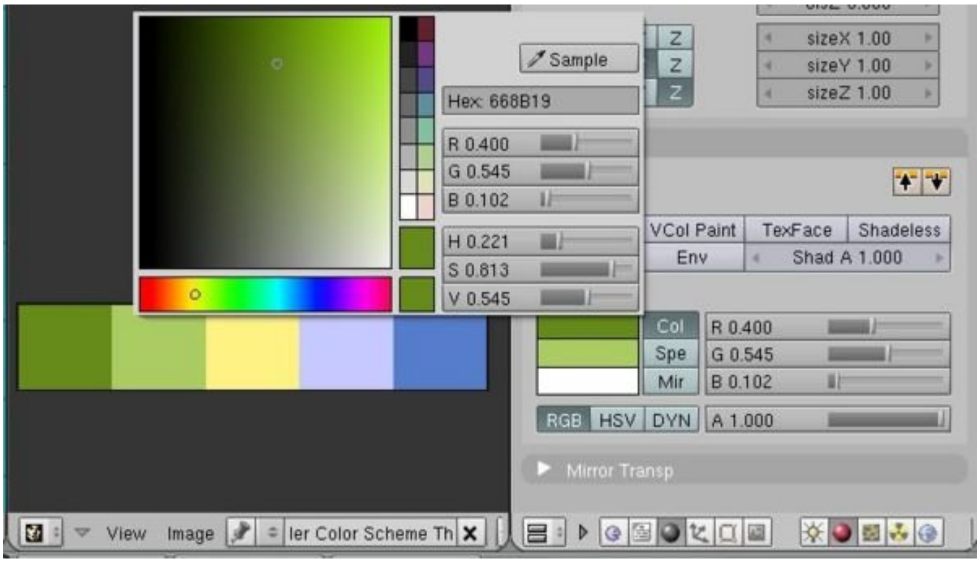
Using each of these panels is discussed in this section. The preview panel attempts to show you what the shader will produce for different kinds of geometric basic shapes. Depending on your buttons window layout, some panels may be tabbed under others, or collapsed.

Sharing Materials

Blender is built to allow you to reuse *anything*, including material settings, between many objects. Instead of creating duplicate materials, you can simply re−use an existing material. There are two ways to do this:

- With the mesh selected, click the up−down selector arrow located to the left of the Material name. The popup list shows you all the current materials. To use one, just click on it.
- In the 3D View, with Ctrl L you can quickly link all selected objects to the material (and other aspects) of the active object. Very useful if you need to set a large number of objects to the same material; just select all of them, then the object that has the desired material, and Ctrl L link them to that "parent".

Color Management with the Color Picker Applet



The term Color Management refers to using a consistent color theme, and specific set of colors, that complement each other and work well to convey the emotional intent of the graphic. There are a number of sites and tools that provide complementary color themes. Choose a theme, and save the pallette (a grid or srip of color swatches that make up the theme) and save it on your computer. In the example above, an ocean/organic theme of five colors, from green through yellow to a blue, are being used. The dark green is being used as a material diffuse color, while a lighter shade is used as the specular color.

To use a color theme, load the pallette image in the UV/Image Editor. Then, when you want to choose a color for a lamp, material, texture, world background, etc, simply:

1. On any Blender panel where a color is chosen, you will see a color swatch that reflects the current RGB settings. The example shows the material settings, where the diffuse *Col*or is being set.
2. Click that swatch, and a color picker applet will pop−up as shown.
3. The applet shows you the currently selected color above the original color, one above the other. left of the RGB sliders.
4. Click the eyedropper *Sample* button, and your cursor will change to an eyedropper.
5. Move your cursor over the desired color in the UV/Image Editor window and click
6. The eyedropper will "suck up" that color and set the current color (top swatch in the color picker applet window) to the new color.
7. Click again to confirm the color

It also might be acceptable, within a color scheme, to change the saturation or value of a color hue. To do so, after picking the hue, to alter the saturation or value of a hue, click on the HSV button, and the RGB sliders will change to HSV sliders.

Changing the *saturation* balances the color against a neutral gray, as if more of the dye was allowed to saturate and permeate the material. The easy way I remember *value* is to think of the dilution of that ink in water; less value means that fewer drops of the ink are in the water, and the dye is therefore not as potent.

Dropper samples a Pixel, not an image area average:

the dropper (when used to sample a picture/image (e.g. in the UV/Image Editor window) or image−textured object may seem to return random colors. This is because the dropper samples a PIXEl, which in a photo image, you can have very different colors next to one another. Mousewheel up in your UV window to zoom in on your image and you will see that, for example, a red pixel next to a green pixel may look yellow at a lower resolution.

Setting a Custom Color Scheme

In the above image, you will notice a vertical strip consisting of 16 mini−swatches, of 8 colors in 2 columns. By default, the left column is shades of gray, and the right column are some basic colors. If you LMB click on one of these, that color will become the current color. If you Ctrl LMB click on one of these, the color swatch will change to become the current color. Ctrl LMB click to save your color scheme this way.

These are session presets and remain active while Blender is running, even if you change files. If you re−start Blender, they reset back to original.

Previous: Manual/Materials	Contents	*Next:* Manual/Material Preview

User Manual: Contents | Guidelines | Blender Version **2.42**

Material Preview

Mode: All Modes

Panel: Shading/Material Context Preview

Hotkey: F5

Description

The Preview panel gives a quick visualisation of the active material and its properties, including its Shaders, Ramps, Mirror Transp properties and Textures. It provides many useful shapes that are very useful for designing new shaders: for some shaders (like those based Ramp colors or a Diffuse shader like Minnaert), one needs fairly complex or specific previewing shapes to decide if the shader−in−design achieves its goal.

Options

Flat XY plane
: Useful for previewing textures and materials of flat objects, like walls, papers and such.

Sphere
: Useful for previewing textures and materials of sphere−like objects, but also to design metals and other reflective/transparent materials, thanks to the checkered background.

Cube
: Useful for previewing textures and materials of cube−like objects, but also to design procedural textures. Features a checkered background.

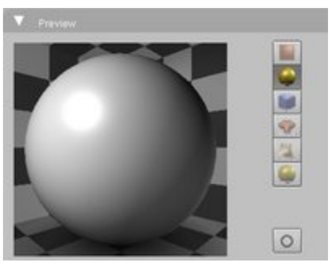
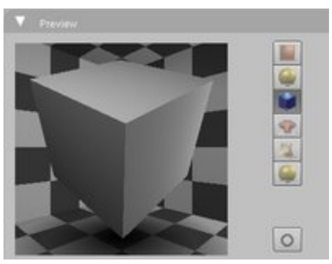
Monkey
: Useful for previewing textures and materials of organic or complex non−primitive shapes. Features a checkered background.

Hair strands
: Useful for previewing textures and materials of strand−like objects, like grass, fur, feathers and hair. Features a checkered background.

Large Sphere with Sky
: Useful for previewing textures and materials of sphere−like objects, but also to design metals and other reflective materials, thanks to the gradient Sky background.

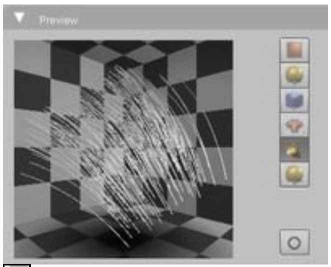
Preview uses OSA (oversampling)
: Whatever the preview option, it will make use of OSA (oversampling) in order to provide with a better quality. Disable this option if your computer is already slow or old.

Examples

Plane preview. Sphere preview. Cube preview.

Monkey preview.



Hair Strands preview.

Sky Sphere preview.

Previous: Manual/Materials in practice Contents *Next:* Manual/Diffuse Shaders

User Manual: Contents | Guidelines | Blender Version **2.43**

Diffuse Shaders

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5

Description

A diffuse shader determines, simply speaking, the general colour of a material when light is shined on it. Most shaders that are designed to mimic reality give a smooth falloff from bright to dark from the point of the strongest illumination to the shadowed areas, but Blender also has other shaders for various special effects.

Options

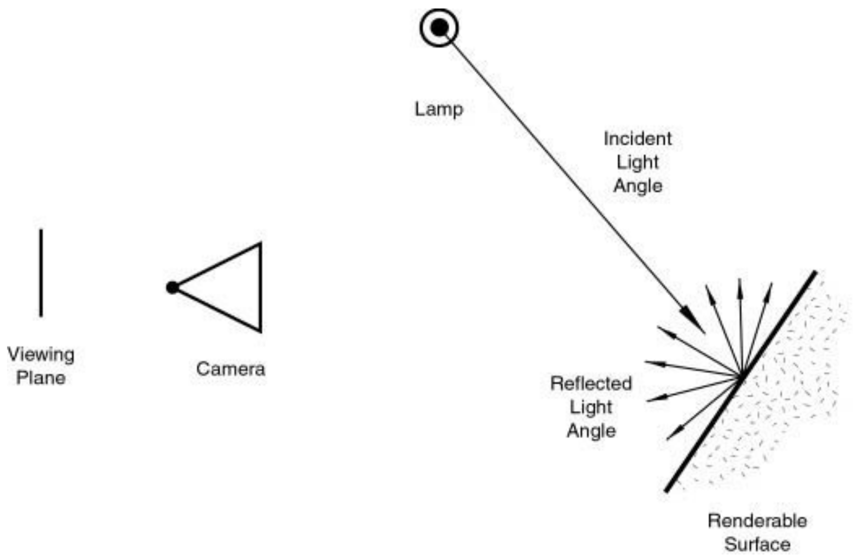
All diffuse shaders have the following options:

Color
: The base *Diffuse color* of the material

Ref
: The shader's brightness, or more accurately, the amount of incident light energy that is actually diffusely reflected towards the camera.

Technical Details

Light striking a surface and then re−irradiated via a Diffusion phenomenon will be scattered, i.e., re−irradiated in all directions isotropically. This means that the camera will see the same amount of light from that surface point no matter what the *incident viewing angle* is. It is this quality that makes diffuse light *viewpoint independent*. Of course the amount of light that strikes the surface depends on the incident light angle. If most of the light striking a surface is reflected diffusely, the surface will have a matte appearance (*Light re−irradiated in the diffusion phenomenon.*).



Light re−irradiated in the diffusion phenomenon.

Hints

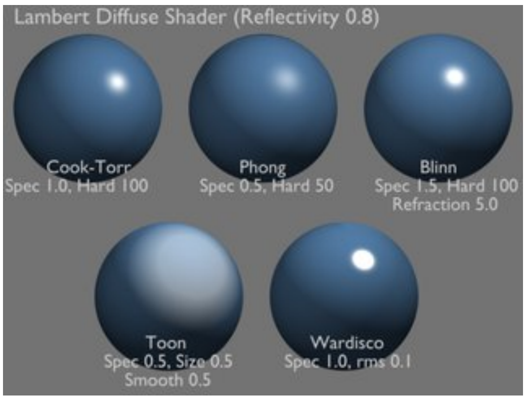
Some shaders' names may sound odd − they are traditionally named after the people who invented them.

Lambert

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5

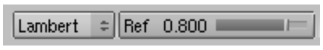


Lambert Shader

Description

This is Blender's default diffuse shader, and is good general all−around work horse...neigh!

Options



The Lambert diffuse shader settings.

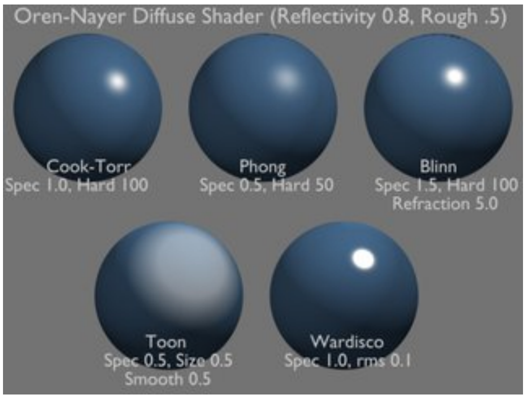
This shader has only the default option, determining how much of available light is reflected. Default is 0.8, to allow some other objects to be brighter.

Oren−Nayar

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5

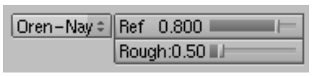


Oren−Nayer Shader

Description

Oren−Nayar takes a somewhat more 'physical' approach to the diffusion phenomena as it takes into account the amount of microscopic roughness of the surface.

Options



The Oren−Nayar diffuse shader settings.

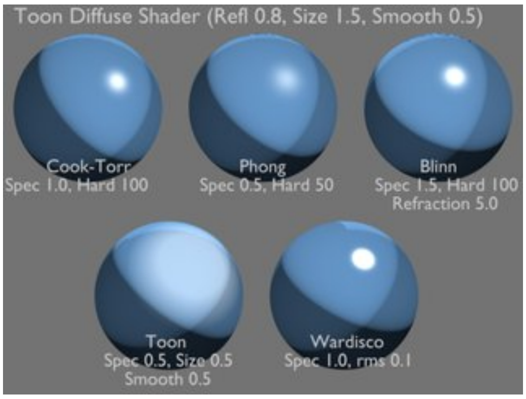
Rough
: The roughness of the surface, and hence, the amount of diffuse scattering

Toon

Mode: All Modes

Panel: Shading/Material Context Shaders

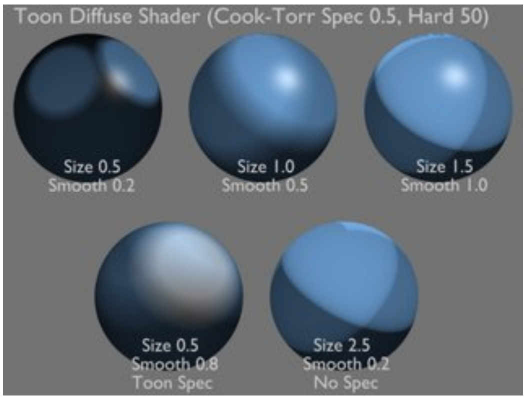
Hotkey: F5



Toon Shader, Different Spec

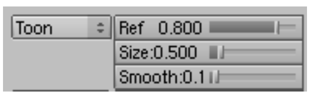
Description

The Toon shader is a very 'un−physical' shader in that it is not meant to fake reality but to produce cartoon cel styled rendering, with clear boundaries between light and shadow and uniformly lit/shadowed regions.



Toon Shader Variations

Options



The Toon diffuse shader settings.

Size
: The size of the lit area

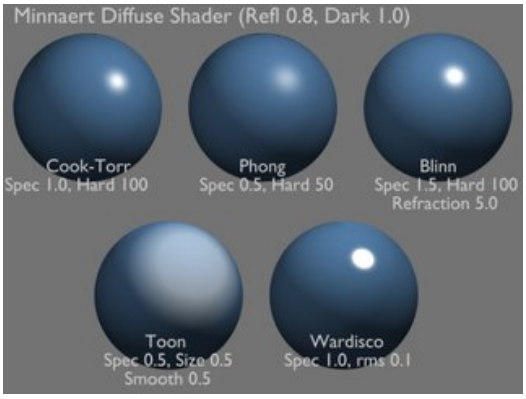
Smooth
: The softness of the boundary between lit and shadowed areas

Minnaert

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5

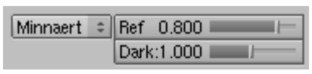


Minnaert Shader

Description

Minnaert works by darkening parts of the standard Lambertian shader, so if *Dark* is 1 you get exactly the Lambertian result. Higher darkness values will darken the center of an object (where it points towards the viewer). Lower darkness values will lighten the edges of the object, making it look somewhat velvet.

Options



The Minnaert diffuse shader settings.

Dark

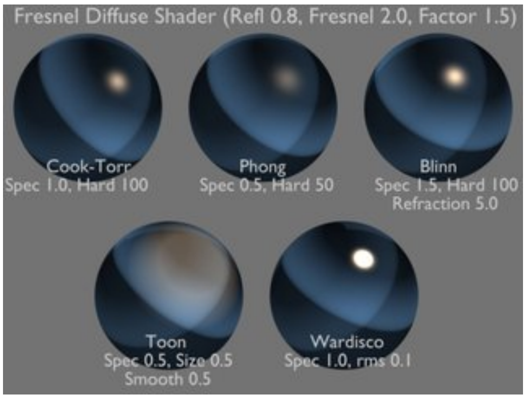
The darkness of the 'lit' areas (higher) or the darkness of the edges pointing away from the light source (lower).

Fresnel

Mode: All Modes

Panel: Shading/Material Context Shaders

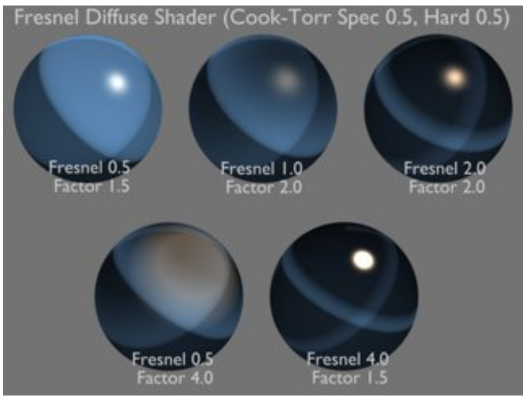
Hotkey: F5



Fresnel Shader, Different Spec

Description

With a Fresnel Shader the amount of diffuse reflected light depends on the incidence angle, i.e. from the direction of the light source. Areas pointing directly towards the light source appear darker, areas perpendicular to the incoming light become brighter.



Fresnel Shader, Same Spec

Options

Ref is the Reflectivity; amount of color reflected for each unit of light received. Fresnel is the power of the Fresnel effect, and Fac is the amount of the effect to blend in.

Previous: Manual/Material Preview | Contents | *Next:* Manual/Specular Shaders

User Manual: Contents | Guidelines | Blender Version **2.43**

Specular Shaders

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5

Description

Specular shaders create the bright highlights that one would see on a glossy surface, mimicking the reflection of light sources. Unlike diffuse shading, specular reflection is *viewpoint dependent*. According to Snell's Law, light striking a specular surface will be reflected at an angle which mirrors the incident light angle (with regard to the surface's normal), which makes the viewing angle very important.

Note

It is important to stress that the *specular reflection* phenomenon discussed here is not the reflection we would see in a mirror, but rather the light highlights we would see on a glossy surface. To obtain true mirror−like reflections you would need to use the internal raytracer. Please refer to section RENDERING of this manual.

Options

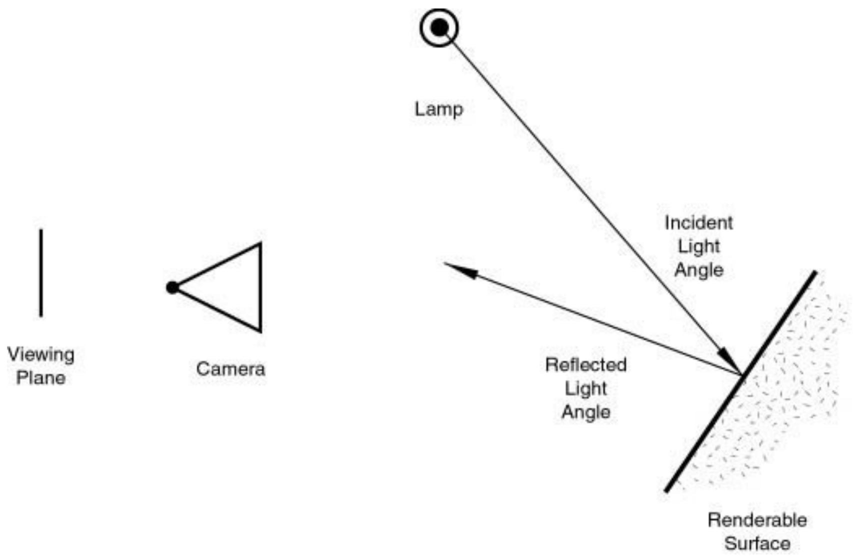
Each specular shader shares two common options:

Specular colour
: The colour of the specular highlight

Spec
: The intensity, or brightness of the specular highlight. This has a range of [0−2], which effectively allows more energy to be shed as specular reflection as there is incident energy.

As a result, a material has at least two different colours, a diffuse, and a specular one. The specular color is normally set to pure white, but it can be set to different values for various effects.

Technical Details



Specular Reflection.

In reality, Diffusion and Specular reflection are generated by exactly the same process of light scattering. Diffusion is dominant from a surface which has so much small−scale roughness in the surface, with respect to wavelength, that light is reflected in many different directions from each tiny bit of the surface, with tiny changes in surface angle.

Specular reflection, on the other hand, dominates on a surface which is smooth, with respect to wavelength. This implies that the scattered rays from each point of the surface are directed almost in the same direction, rather than being diffusely scattered. It's just a matter of the scale of the detail. If the surface roughness is much smaller than the wavelength of the incident light it appears flat and acts as a mirror.

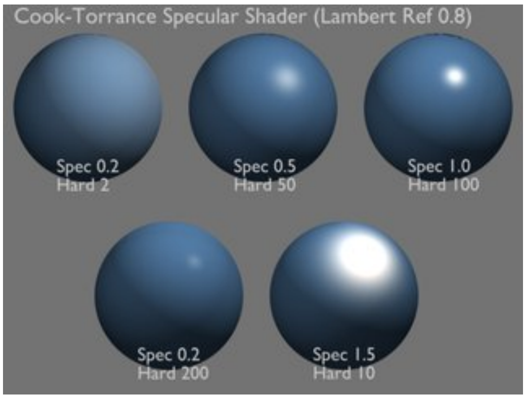
If it is difficult to you to understand the relation between roughness' scale and light's wavelength, try to imagine a ball (say, of centimetre scale): if you throw it against a wall of raw stones (with a scale of roughness of a decimetre), it will bounce in a different direction each time, and you will likely quickly lost it! On the other hand, if you throw it against a concrete wall (with a roughness of, say, a millimetre scale), you can quite easily anticipate its bounce, which follow (more or less!) the same law as the light reflectionâ€¦

CookTorr

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5



Cook−Torrance Shader

Description

CookTorr (Cook−Torrance) is a basic specular shader that is most useful for creating shiny plastic surfaces. It is a slightly optimised version of Phong.

Options

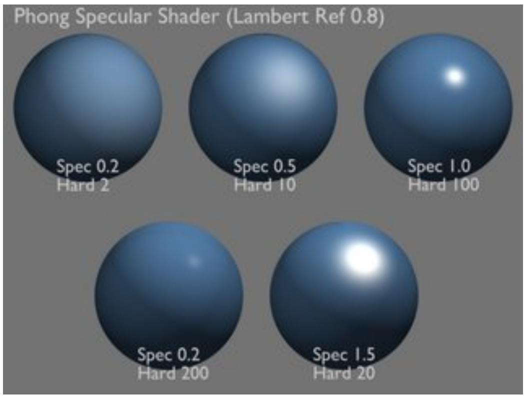
Hard
: The hardness, or size of the specular highlight

Phong

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5



Phong Shader

Description

Phong is a basic shader that's very similar to CookTorr, but is better for skin and organic surfaces

Options

Hard
: The hardness, or size of the specular highlight

Planet Atmosphere

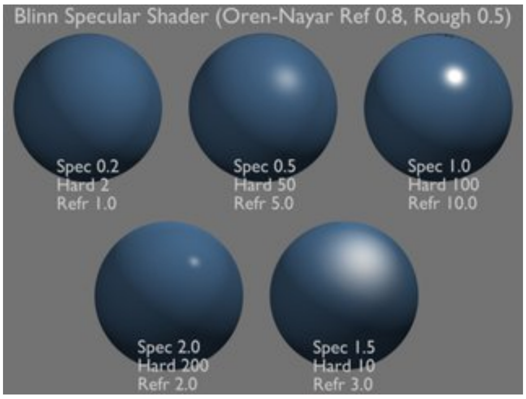
Because of its fuzzyness, this shader is good for atmosphere around a planet. Add a sphere around the planet, slightly larger than the planet. For its material, use a phong specular shader. Set it very low alpha (.05), zero diffuse, low hardness (5) but high specularity (2).

Blinn

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5



Blinn Shader

Description

Blinn is a more 'physical' specular shader, often used with the Oren−Nayar diffuse shader. It can be more controllable because it adds a fourth option, an *index of refraction*, to the aforementioned three.

Options

Hard

The hardness, or size of the specular highlight. The Blinn shader is capable of much tighter specular highlights than Phong or CookTorr.

Refr

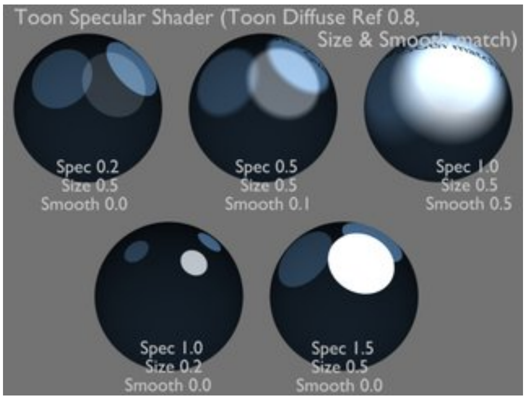
The 'Index of Refraction'. This parameter is not actually used to compute refraction of light rays through the material (a ray−tracer is needed for that), but to correctly compute specular reflection intensity and extension via Snell's Law.

Toon

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5



Toon Specular Shader

Description

The Toon specular shader matches the Toon diffuse shader. It is designed to produce the sharp, uniform highlights of cartoon cels.

Options

Size
: The size of the specular highlight

Smooth
: The softness of the highlight's edge

Hints

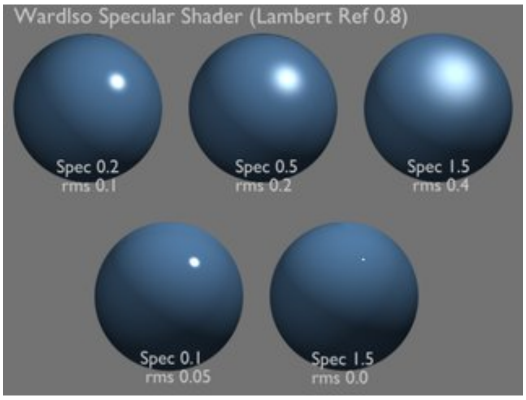
Toon shader can be also be accomplished in a more controllable way using ColorRamps.

WardIso

Mode: All Modes

Panel: Shading/Material Context Shaders

Hotkey: F5



WardIso Shader

Description

WardIso is a flexible specular shader that can be useful for metal or plastic.

Options

rms
: rms controls, in effect, the size of the specular highlight, though using a different method to that of the other specular shaders. It is capable of extremely sharp highlights.

Previous: Manual/Diffuse Shaders Contents *Next:* Manual/Material Options

User Manual: Contents | Guidelines | Blender Version **2.43**

Materials

Mode: All Modes

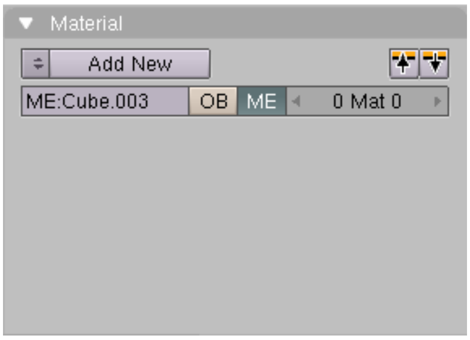
Panel: Shading/Material Context Material

Hotkey: F5

Description

Materials can be linked to objects and obData in the materials panel, of the Shading/Material context. Here is where you can manage how materials are linked to objects, meshes, etc. and activate a material for editing in the rest of the panels.

Options



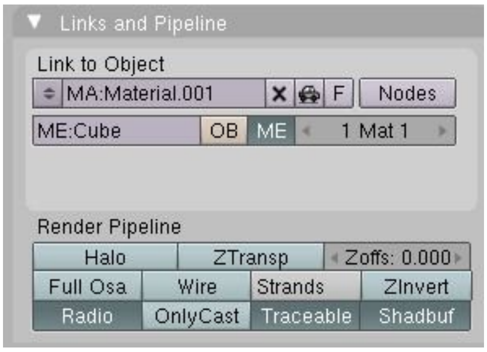
Material Tab

If there are no materials linked to the active object, the following options are available:

Add New
: Add a new material and link it to the active object or object data. Like other datablocks, Blender will automatically set its name to Material.001 and so on. It's a very good idea to give your materials clear names so you can keep track of them, especially when they're linked to multiple objects.

Select an existing material ( )
: Choose an existing material from a list. If there are LOTS of materials, you will see a choice "DataSelect". Click that and one of your windows will change to a data browser window, listing all of the materials for you to choose from.

Links and Pipeline Options



Links and Pipeline Tab

With a material linked or created, further options are available:

MA
: (Material name, 

)
: Shift−click into this field to name your material

Number of users (number field)
: The number of objects or obdata that use the material. This material is linked between the various objects, and will update across all of them when edited. Clicking this number will make a 'single user copy', duplicating the material, with it linked only to the active object/obdata.

Auto ()
: Automatically generates a (hopefully relevant) name for the material, based on the diffuse colour

F (Fake user)
: Gives the material a 'fake user', to keep the material datablock saved in the .blend file, even if it has no real users

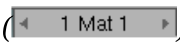
Nodes
: Designates this material to be a material node noodle, and not from the Material/Ramps/Shaders settings.

*Datablock links (

)*
: These two buttons determine whether the material is linked to the object or to (in this case) the mesh (or curve, nurbs, etc.). The ME button determines that this material will be linked to the mesh's datablock which is linked to the object's datablock. The OB button determines that the material will be linked to the object's data block directly.

  This has consequences of course. For example, different objects may share the same mesh datablock. Since this datablock defines the shape of the object any change in edit mode will be reflected on all of those objects. Moreover, anything linked to that mesh datablock will be shared by every object that shares that mesh. So, if the material is linked to the mesh, every object will share it.

  On the other hand, if the material is linked directly to the object datablock, the objects can have different materials and still share the same mesh. Short explanation: If connected to the object, you can have several instances of the same obData using different materials. If linked to mesh data, you can't.

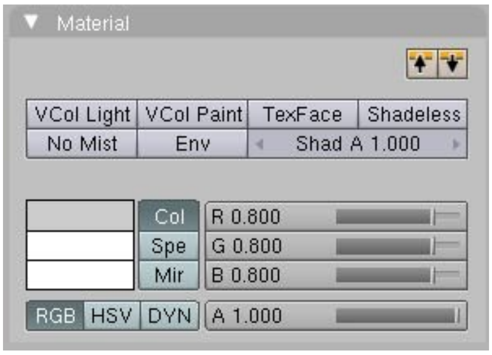
*Material indices (

)*
: This shows how many materials you have for that obData and which one is currently active for editing. You can have multiple materials on an object and this can be done in the Editing Context (F9) in the Link and Materials Tab. See [Manual/Multiple Materials] for more info.

Render Pipeline
: These toggles tell Blender where this material fits into the Render Pipeline, and what aspects of the material are to be rendered.
    - ◊ Halo − Each vertex is a halo of light

◊ ZTransp − Objects behind this one can shine through it
◊ Zoffs: A numeric input for offsetting the z−value; negative numbers puts this surface just in front of others assigned to the same mesh
◊ Full OSA − perform full over−sampling on this material to make it as smooth as possible where it overlays or intersects with other materials.
◊ Wire − render it in wireframe mode
◊ Strands−Colors static particles (hair, fur)
◊ ZInvert−renders faces inside−out. With OSA on, the render of one object against the backdrop of another object may occur *after* the render of the first object, and therefore the OSA calculation may pick up a halo or outline of the original backdrop. Use this option to disable this.
◊ Radio−material participates in Radiosity baking
◊ Only Cast − Material does not show up as a color, but just casts shadows onto other objects.
◊ Traceable−participates in ray tracing, if enabled during render
◊ Shadbuf−Can participate in shadow buffering, a faster alternative to raytracing

Material Options



Material Tab

Copy / Paste ( *)*
: Copies the active material's settings / Pastes stored material settings into the active material

Modes
: VCol Light: kinda like Halo, but each vertex's color is used as a light source
VCol Paint: enables [Vertex Paint]
TexFace: colors are taken from a [UV Texture]
Shadeless − color is not affected by light or shadows
No Mist − color is not obscured or faded by world mist
Env − renders it invisible
Shad A − this numeric control sets the Alpha value of shadows cast by this mesh, thus softening them

Colors
: There are three colors to set:
◊ Col − [Diffuse Color]
◊ Spe − [Specular Color]
◊ Mir − [Mirror Color].
Clicking a color swatches brings up the color picker applet, which allows you to pick a pre−defined standard color, or click on the eyedropper to sample a color, or play with the RGB sliders, or play with the HSV box and click on a color inside the gradient. The RGBA sliders set the active color. Click Col, Spe, or Mir to activate a color settings for the sliders to set.

Color Expression
: RGB is for Red Green Blue sliders;
HSV changes the sliders to set Hue Saturation Value

Dynamics

For the Game Engine, these set up friction for the surface

Previous: Manual/Specular Shaders	Contents	*Next:* Manual/Ramps

User Manual: Contents | Guidelines | Blender Version **2.43**

Ramps

Mode: All Modes

Panel: Context Shading â†' sub−context Material â†' Ramps

Hotkey: F5

In many real life situations − like skin or metals − the colour of diffuse and specular reflections can differ slightly, based on the amount of energy a surface receives or on the light angle of incidence. The new Ramp Shader options in Blender now allow you to set a range of colours for a Material, and define how the range will vary over a surface, and how it blends with the 'actual colour' (typically from material or as output of texture).

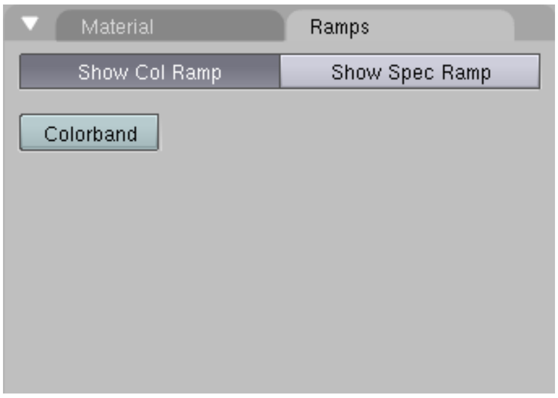
Description

Ramps allow you to precisely control the colour gradient across a material, rather than just a simple blend from a brightened colour to a darkened colour, from the most strongly lit area to the darkest lit area. As well as several options for controlling the gradient from lit to shadowed, ramps also provide 'normal' input, to define a gradient from surfaces facing the camera to surfaces facing away from the camera. This is often used for materials like some types of metallic car paint, that change colour based on viewing angle.

Since texture calculations in Blender happen before shading, the Ramp Shader can completely replace texture or material color. But by use of the mixing options and Alpha values it is possible to create an additional layer of shading in Blender materials.

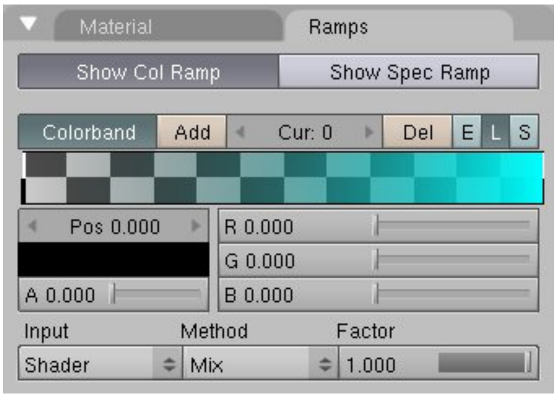
Options

The Ramps panel is located in the Material context (F5). Here you can use the top two buttons to show either settings by pressing Show Col Ramp for diffuse or Show Spec Ramp for specular ramps (*Ramps Panel.*).



Ramps Panel.

Pressing the button Colorband enables Ramp Shaders. By default it opens with two colours, with the first having **Alpha = 0**, and no colour. And the second having **Alpha = 1** and a cyan colour (*Ramps Panel Colorband.*).



Ramps Panel Colorband.

See the settings description of the colorband below.

The two pop−up buttons and the value slider in the bottom of the panel defines how the Ramp Shaders work:

Input popup menu.

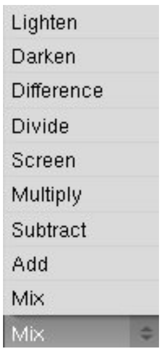
Input
: The input menu contains the following options for defining the gradient:

  Shader
  : The value as delivered by the material's shader (like Lambert or Phong) defines the colour. Here the amount of light doesn't matter for colour, only the direction of the light.

  Energy
  : As Shader, but now also lamp energy, colour and distance, is taken into account. This makes the material change colour when more light shines on it.

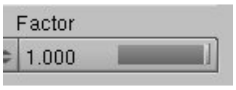
  Normal
  : The surface normal, relative to camera, is used for the Ramp Shader. This is possible with a texture as well, but added for convenience.

  Result
  : All three previous options work per lamp, this option only does it in the end of all shading calculation. This allows full control over the entire shading, including 'Toon' style results. Using alpha values here is most useful for tweaking a finishing touch to a material.



Method popup menu.

Method
: A list of the various blending modes available for blending the ramp shader with the colour from Input.



Factor slider.

Factor
: The Factor slider denotes the overall factor of the Ramp Shader effect: **0.0** means no effect and **1.0** means full effect.

Colorbands

Mode: All Modes

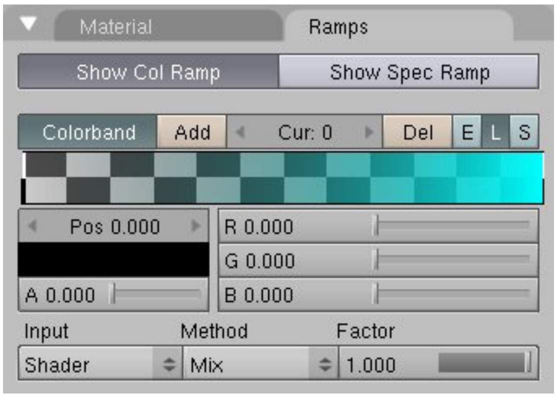
Panel: Context Shading â†’ sub−context Material â†’ Ramps

Hotkey: F5

Description

A colorband can contain a gradient through a sequence of many colours (with alpha), each colour acting across a certain position in the spectrum. Colorbands are used in both materials and textures, as well as other places where a range of colours can be computed and displayed.

Options



Ramps Panel Colorband.

Add

Add a new mark to the centre of the colorband with default colours (neutral grey). New marks can also be added with Ctrl LMB click on the colorband itself, which will add the mark at the position of the click, with the same colour as already existing underneath the mouse pointer.

Del
: Remove the current mark from the colorband.

Cur
: The number of the current selected mark on the colorband that's being edited. The current selected mark is shown on the colorband with double width. To select a colour mark you can either press LMB on the desired one, or step the current colour number up and down with the right and left arrows in this control. You can also Shift LMB in the field and enter the required colour number manually.

Pos
: The location of the current mark along the colorband, from **0.0** (left) to **1.0** (right). The position of the marks can also be changed by clicking and dragging them with LMB.

Reordering Colours

If you reorder the positions of the colours, they will be renumbered so that they always start with **0** from the left and increment to the right.

R/G/B
: The colour of the current mark. You can LMB on the colour swatch under the Pos field to choose a colour using the Color Picker.

Alpha
: The alpha value (opacity) of the current mark. An Alpha value of **0.0** means that the colour is totally transparent and will not be seen in the final colorband. A value of **1.0** sets the colour opaque. If you defined colours with different Alpha values, they will be interpolated between each other to get a smooth transition between different transparency settings. You can preview the Alpha settings on the colorband with the checker board pattern behind the colorband. If pattern is visible then the transparency is less than **1.0**.

The colours can interpolate from one to the other in the following ways:

E
: Ease in by a quadratic equation.

C
: Cardinal.

L
: Linear (Default). A smooth, consistent transition between colours.

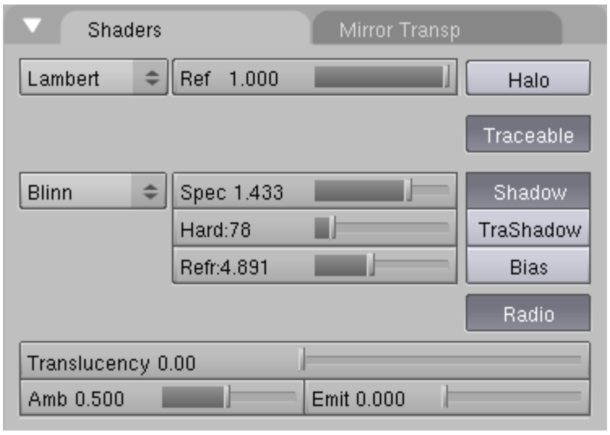
S
: B−Spline.

Examples

In this first example to the right, we want to texture a snake, specifically the deadly coral snake. We want to make a repeating set of four colours: black, yellow, red, yellow (and then back to black again). We also want to make the rings sharp in definition and transition. This example uses **8** colorband settings: **0** and **7** are black; **1** and **2** are yellow, **3** and **4** are red, and **5** & **6** are yellow. Position **0** and **1** close together, **2** and **3**, etc. Use a little noise and turbulence; together with the scales texture you should get really close!

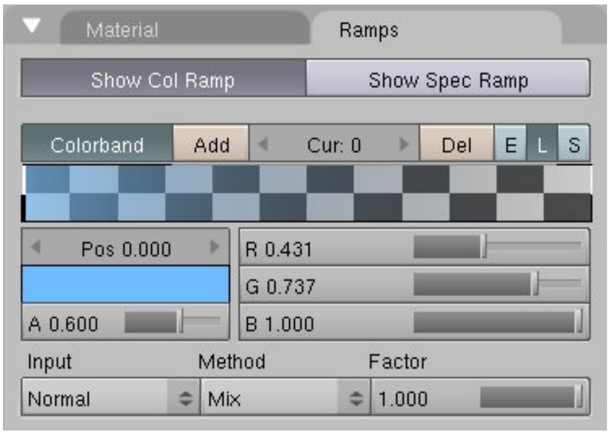
Let us make another simple example using Ramp Shaders.

- Remove default cube object from scene and add a Monkey mesh (Shift A â†’ Add â†’ Mesh â†’ Monkey). Press Subsurf (NOTE: In 2.42 and higher, Subsurf is now in the Modifier panels), and set display and render Subsurf Subdivision level to **2**. Press Set Smooth to get a nice smooth Monkey. All this in Edit context (F9).



Shader settings.

- Now press TAB to exit Edit mode. Press F5 to enter Material context. In the Material panel press Add New to add a new material. Change the parameters in the Shaders tab as (*Shader settings*).

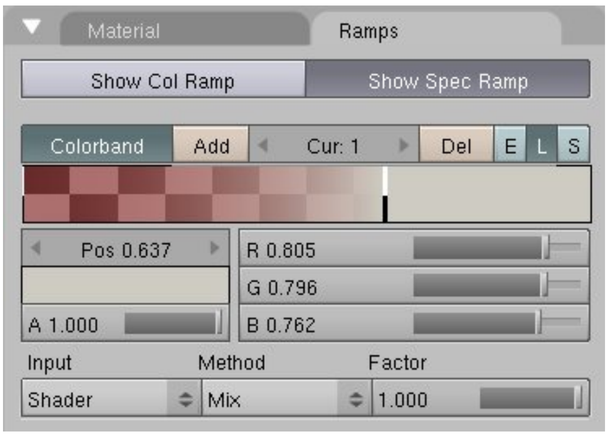


Ramp Shader settings.

- Press Ramps tab to open the Ramp Shader panel. Press Colorband to activate the Ramp Shader effect. Now try to set the parameters as (*Ramp Shader settings*). Remember to set the Input to Normal. The second colour to the right is set to Alpha = **0.0** and the colour to pure black.

Specular Ramp Shader color 0.

- In the Ramps tab press Show Spec Ramp and try to set the parameters as (*Specular Ramp Shader color 0*) and (*Specular Ramp Shader color 1*).



Specular Ramp Shader color 1.

Here is the rendered result of the settings we just entered above. In (*No Ramp Shader*) there is no Ramp Shader active. In (*Color Ramp*) the Color Ramp is activated and finally in (*Both Color and Specular Ramp*) both Color Ramp and Specular Ramp are activated! Remember here we have just demonstrated one effect of the Ramp Shader. There is much more to explore, try changing the Input and Method parameters, to see totally different results from the ones we have just seen in this example.

No Ramp Shader.

Colour Ramp.

Both Colour and Specular Ramp.

Previous: Manual/Material Options | Contents | *Next:* Manual/Raytraced Reflections

User Manual: Contents | Guidelines | Blender Version **2.46**

This page is being updated IN ANTICIPATION of some of the changes expected in the NEXT release of Blender.

Specific featurees and options discussed may not be available in the current release.

Raytraced Mirror Reflections

Mode: All Modes

Panel: Shading/Material Context Mirror Transp

Hotkey: F5

Description

Raytracing can be used to make a material reflect its surroundings, like a mirror. The principle of raytraced reflections is very simple: a ray is fired from the camera and travels through the scene until it encounters an object. If the first object hit by the ray is not reflective, then the ray takes the color of the object. If the object is reflective, then the ray bounces from its current location and travels up to another object, and so on, until a non−reflective object is finally met and gives the whole chain of rays its color.

Eventually, the first reflective object inherits the colors of its environment, proportionally to its RayMir value. Obviously, if there are only reflective objects in the scene, then the render could last forever. This is why a mechanism for limiting the travel of a single ray has been set through the Depth value: this parameter sets the maximum number of bounces allowed for a single ray.

Note

You need to enable raytracing in your scene settings if you want to use raytraced reflections. This is done in the Scene/Render context Render Panel. Raytracing is enabled by default in Blender 2.37 and higher.

The *Mir* setting on the Material panel is the color of the light reflected back. Usually, for normal mirrors, use White. However, some mirrors color the reflection, so you can change the color by clicking on the swatch on that panel. The Mirror Transp panel determines how and whether the mirror actually is active and reflects things. If you set RayMir to something >0, then the mirror is active. The reflection will be the tinted the color you set on the Material panel.

Options

The Mirror Transp Panel.

Ray Mirror
: Enables and disables raytraced reflections

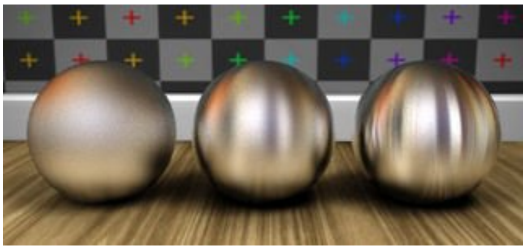
RayMir
: Sets the amount of reflectiveness of the object. Use a value of 1.00 if you need a perfect mirror, or set RayMir to 0.00 if you don't want any reflection.

Fresnel
: Sets the power of the Fresnel effect. The Fresnel effect controls how reflective the Material is, depending on the angle between the surface normal and the viewing direction. Typically, the larger the angle, the more reflective a Material becomes (this generally occurs on the outline of the object).

Fac
: A controlling 'factor' to adjust how the blending (between reflective and non−reflective areas) happens.

Suzanne in the Fun House

Gloss
: In paint, a high−gloss finish is very smooth and shiny. A flat, or low gloss disperses the light and gives a very blurry reflection. Also, uneven or waxed−but−grainy surfaces (such as car paint) are not perfect and therefore slightly need a Gloss < 1.0. In the example to the right, the left mirror has a Gloss of 0.98, the middle is Gloss = 1.0, and the right one has Gloss of 0.90. Use this setting to make a realistic reflection, all the way up to a completely foggy mirror. You can also use this value to mimic depth of field in mirrors.



Anisotropic tangent reflecting spheres with anisotropy of 0.0, 0.75, and 1.0

Anistropy
: The amount a reflection is stretched in the tangent direction. If the tangent shading option is on, Blender automatically renders blurry reflections as anisotropic reflections.

When Tangent is switched on, the *Aniso* slider controls the strength of this anisotropic reflection, with a range of 1.0 (default) being fully anisotropic and 0.0 being fully circular, as is when tangent shading on the material is switched off. Anisotropic raytraced reflection uses the same tangent vectors as for tangent shading, so you can modify the angle and layout the same way, with the auto−generated tangents, or based on the mesh's UV co−ordinates.

Samples
: The number of samples to average to arrive at the pixel's final color. More samples will give a smoother result, but the greater the number of samples, the slower the render speed.

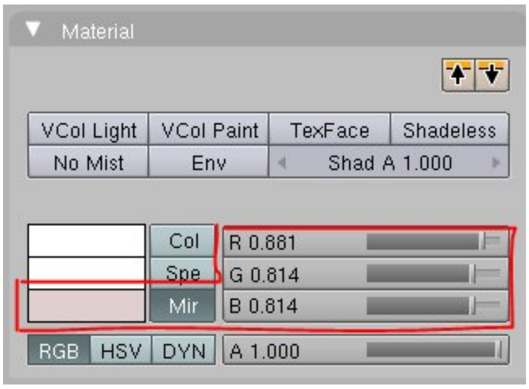
Threshold
: The threshold for adaptive sampling. Sampling is skipped when no more samples are deemed necessary, by checking the statistical variance of the samples so far for that pixel against the threshold. Raising the threshold will make the adaptive sampler skip more often, however the reflections could become noisier.

Depth
: Sets the maximum number of bounces for a single ray to be reflected. The default Depth of 2 is typically a good value. If your scene contains many reflective objects and/or if the camera zooms in on such a reflective object, you will need to increase this value if you want to see surrounding reflections in the reflection of the reflected object (!). In this case, a Depth of 4 or 5 is typically a good value.

Max Distance
: The number of blender units away from camera (Z−Depth) beyond which to stop calculating the actual reflection and to simply use the fade−out method. In the example, there is a mirror behind the camera, 10 blender units from the middle mirror. The top reflection of Suzy is actually a reflection of the reflection off the back mirror. The front mirror has Max Distance set to 20, so, as you can see, it is starting to fade to sky color.

Ray−End Fade Out
: For objects that recede away from the camera, further than Max Distance set above, you can have the mirror effect fade out, which reduces compute time. A large reflecting pond, at the far side, just fades out to be the sky color. You have two choices:
    - ◊ Fade to Sky color − Uses the sky color in the world settings
    - ◊ Fade to Material color − uses the material color

Colored reflections

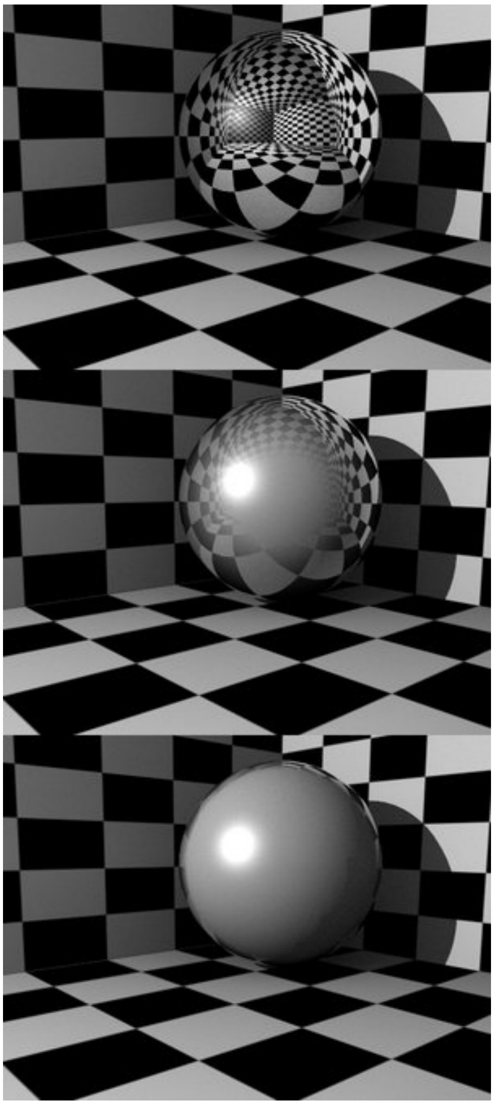


Mirror Color

By default, an almost perfectly reflective Material like Chrome, or a Mirror object, will reflect the exact colors of its surrounding. But some other equally reflective Materials tint the reflections with their own color. This is the case for well polished copper and gold, for example. In order to replicate this within Blender, you have to set the Mirror Color accordingly. In the example above, the middle mirror has the mirror color set as shown to the right.

Examples

Fresnel



Demonstration of Fresnel effect with values equal to (from top to bottom) 0.0, 2.5 and 5.0

Let's undertake a small experiment in order to understand what Fresnel is really about. After a rainy day, go out and stand over a puddle of water. You can see the ground through the puddle. If you kneel just in front of the puddle, your face close to the ground, and look again at a distant point on the puddle of water, the liquid surface part which is closer to you lets you see the ground, but if you move your gaze towards the other end of the puddle, then the ground is gradually masked until all you see is the reflection of the sky. This is the Fresnel effect: having a surface sharing reflective and non−reflective properties according to the viewing angle and the surface normal.

In *Demonstration of Fresnel effect with values equal to (from top to bottom) 0.0, 2.5 and 5.0*, this behavior is demonstrated for a perfectly reflective Material (RayMir 1.0).

Fresnel 0.0 stands for a perfect mirror Material, while Fresnel 5.0 could stand for a glossy Material (varnished wood, for example?). It's barely noticeable but in the lower picture, the Material is perfectly reflective.

Reference

For more information, including other examples and in−depth discussion, please visit:

- Blender improved ray tracing
- QMC Sampling
- Mental Ray info

Hints

In order to get a physically accurate Fresnel effect with the current algorithm, you have to set Fresnel to 5.0 and Fac to 1.25. Nevertheless, you can play with these values for the sake of artistic freedom, if you feel the need to.

Environment Maps (EnvMaps) can also be used to simulate reflective materials. Environment maps are more complicated to set up, have many limitations and are much less accurate, particularly on non−planar surfaces. However, Environment maps can be a lot faster to render and support extra features like filtering the reflection map to fake blurred reflections.

Previous: Manual/Ramps	Contents	*Next:* Manual/Raytraced Transparency

User Manual: [Contents] | [Guidelines] | Blender Version **2.43**

Raytraced Transparency

Mode: All Modes

Panel: Shading/Material Context Mirror Transp

Hotkey: F5

Description

Raytracing is also used for simulating the refraction of light rays through a transparent material, like a lens. A ray is sent from the camera and travels through the scene until it encounters an object. If the first object hit by the ray is non−transparent, then the ray takes the color of the object. If the object is transparent, then the ray continues its travel through it to the next object, and so on, until a non−transparent object is finally encountered which gives the whole chain of rays its color. Eventually, the first transparent object inherits the colors of its background, proportionally to its Alpha value (and the Alpha value of each transparent Material hit in−between). But while the ray travels through the transparent object, it can be deflected from its course according to the Index of Refraction (IOR) of the material. When you actually look through a plain sphere of glass, you will notice that the background is upside−down and distorted: this is all because of the Index of Refraction of glass.

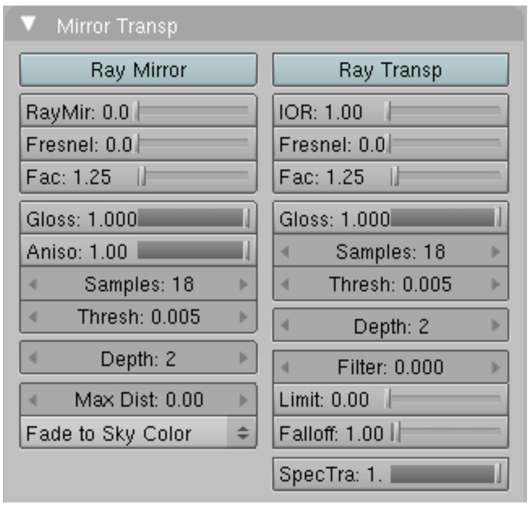
Enable Raytracing

You need to enable raytracing in your scene settings if you want to use raytraced transparency and refraction. This is done in the Scene/Render context Render Panel. Raytracing is enabled by default in Blender 2.37 and higher.

Alpha value

You need to set your Alpha value (A in the Material Pannel) at something else than 1.000. However, the alpha value will still be reported in the render as 1.0

Options



The Mirror Transp Panel.

RayTransp
: Enables and disables raytraced transparency

Filt
: Amount of filtering for transparent ray trace. The higher this value, the more the base color of the material will show. The material will still be transparent but it will start to take on the color of the material.

IOR
: Sets how much a ray travelling through the Material will be refracted, hence producing a distorted image of its background. See Examples.

Depth
: Sets the maximum number of transparent surfaces a single ray can travel through. There is no typical value. Transparent objects outside the Depth range will be rendered pitch black if viewed through the transparent object that the Depth is set for. In other words: if you notice black areas on the surface of a transparent object, the solution is probably to increase its Depth value (this is a common issue with raytracing transparent objects).

Limit
: materials thicker than this are not transparent. This is used to control the threshold after which the filter color starts to come into play.

Falloff
: how fast light is absorbed as is passes through the material. Gives 'depth' and 'thickness' to glass.

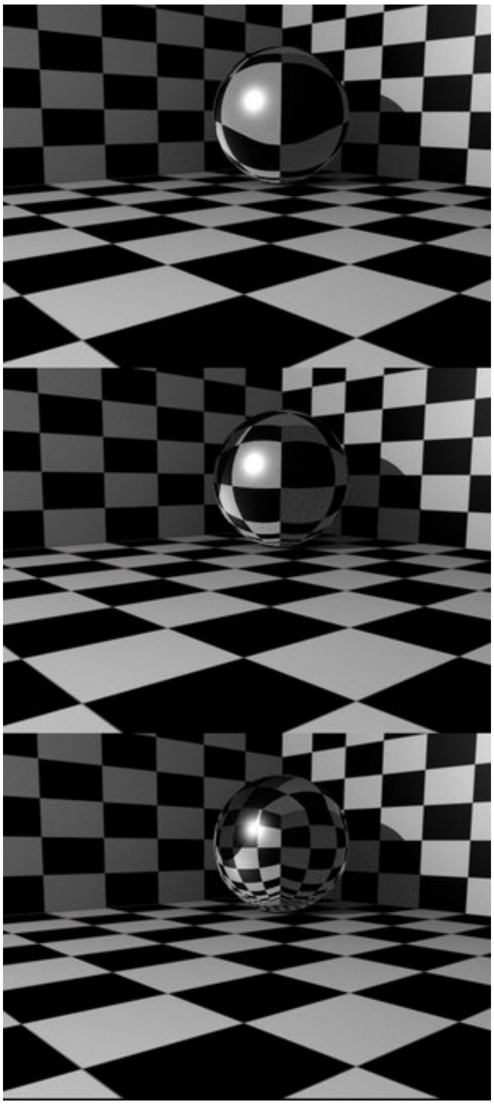
Fresnel
: Sets the power of the Fresnel effect. The Fresnel effect controls how transparent the Material is, depending on the angle between the surface normal and the viewing direction. Typically, the larger the angle, the more opaque a Material becomes (this generally occurs on the outline of the object).

Fac
: A controlling 'factor' to adjust how the blending (between transparent and non−transparent areas) happens.

SpecTra
: This slider controls the Alpha/falloff for Specular colors

Examples

Index of Refraction



Influence of the IOR of an Object on the distortion of the background: spheres of Water, Glass and Diamond (top to bottom).

(*Influence of the IOR of an Object on the distortion of the background: spheres of Water, Glass and Diamond (top to bottom).*). There are different values for typical materials: Air is **1.000** (no refraction), Alcohol is **1.329**, Glass is **1.517**, Plastic is **1.460**, Water is **1.333** and Diamond is **2.417**.

Glass

Realistic Glass is not perfectly transparent, and so light can reflect off of it, it can bend light, and you can even sometimes see shadows cast on the glass, since in reality the non−shadowed surface of the glass is reflecting some light back at you.

Realistic Glass Settings

In the Render panel, enable Raytracing and Shadows. In the example render shown, we have a lamp just off camera to the right, a red−colored ball, an angled piece of glass covering the bottom of the picture, a teal−colored NURBS torus (donut), and a back wall, in Z order. You can see:

- Refracted wall, the donut, the environment behind the glass
- Shadows cast onto the glass by the red ball in front of the glass
- Shadows cast through the glass onto the donut by the red ball
- Shadow cast by the ball behind the glass into the box
- Refraction of the shadow of ball and torus coming back through the glass
- Specular glare on the glass from the lamp

In the Material settings, shown here for slightly blue green glass, enable Traceable and Shadbuff so that Raytraced as well as Shadow Buffer lamps will work with the glass.

In the Material panel, notice that the Alpha is not pure 0.0 In the Shaders panel, Shadow and TraShadow are enabled. The glass is colored by ambient light and emits its color, to simulate light bouncing around inside the glass itself and emerging

In the Mirror/Transp panel, default settings are used except for IOR.

Atmosphere/Cloud Cover


SSS
Ramps
Material
VCol Light
VCol Paint
TexFace
Shadeles
No Mist
Env
Shad A 1.000
Col
Spe
Mir
R 1.000
G 1.000
B 1.000
RGB
HSV
DYN
A 0.000
Shaders
Mirror Transp
Ray Mirror
Ray Transp
RayMir: 0.
Fresnel: 0.
Fac: 1.25
IOR: 1.00
Fresnel: 0.
Fac: 1.25
Gloss: 1.0
Aniso: 0.0
Samples: 18
Thresh: 0.005
Depth: 2
Max Dist: 0.00
Fade to Sky Color
Gloss: 1.0
Samples: 18
Thresh: 0.005
Depth: 2
Filter: 0.000
Limit: 0.00
Falloff: 1.0
SpecTra: 1
Map To
Col
Nor
Csp
Cmir
Ref
Spec
Amb
Hard
RayMir
Alpha
Emit
TransLu
Disp
Stenci
Neg
No RGB
Mix
R 0.418
G 0.393
B 0.651
Col 0.527
Nor 0.50
Var 1.000
Disp 0.200
DVar 1.0
Warp
fac 0.00
Map Input
Texture
Links and Pipeline
Preview

Cloud Cover Settings

In this example, we need a cloud cover for a planet. This is used to texture a UV Sphere that is a larger diameter than the planet inside of it.

The Alpha of the base material is zero, as is any Ambient or Emit values.

Of course, we have to use the Cloud texture, and put it to two uses: to slightly affect the color of the mesh, and to affect the Alpha transparency. While the base material is white, the texture gives it a blue tinge. It receives shadows, so the shadow of the planet will actually darken the dark side of the atmosphere.

Where the texture is black, think 0 color. If you map that 0 to alpha, you get transparent, or see−through. Where the texture is white, think full, 1.0 color. If you map white to alpha, you get full opaqueness. If that 1.0 also maps to some color, like blue, you will get a fully opaque blue color. In the example to the right, that full blue is then mixed down by the Col slider in the texture Map To panel so that it mixes with the base white material and only shades the white a slight hue of blue.

To make the clouds more pronounced, you can map the input to a higher value, and/or multiply (not mix) two texture channels layers together.

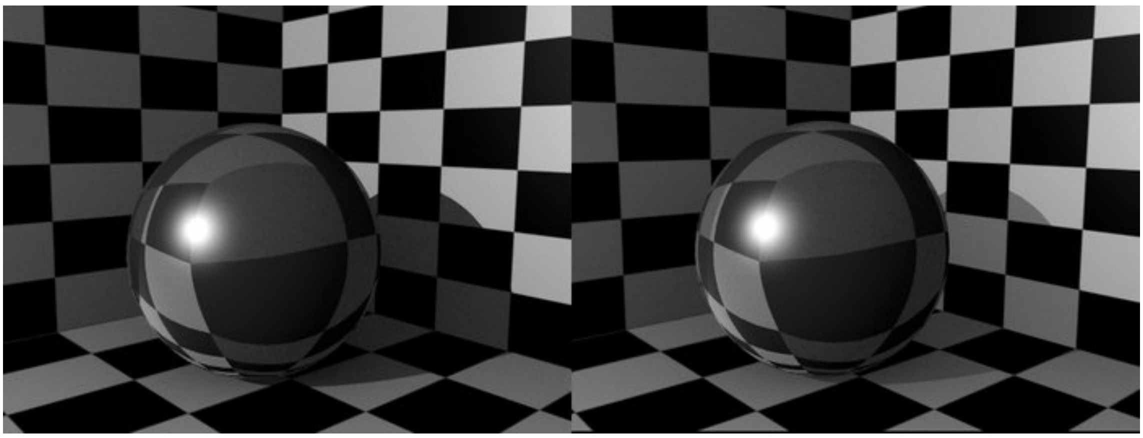
Hints

In order to get a physically accurate Fresnel effect with the current algorithm, you have to set Fresnel to 5.0 and Fac to 1.25. Nevertheless, you can play with these values for the sake of artistic freedom, if you feel the need to.

Casting Transparent Shadows

By default, the shadows of transparent objects are rendered solid black, as if the object was not transparent at all. But in reality, the more transparent an object is, the lighter its shadow will be. This can be taken into account, not in the Mirror Transp panel transparent object settings.

Transparent shadows are set on the materials that **receive** the shadows from the transparent object. This is enabled and disabled with the TraShadow button, in the Shading/Material context Shaders Panel. The shadow's brightness is dependent on the Alpha value of the shadow casting Material.



Casting *transparent* shadows: TraShado 'off' on the left, TraShado 'on' to the right.

Ray versus Z Transparency

ZTransp, located in the Material Links panel, causes Blender to use the Z value, or distance from camera, to bleed things through. When using transparent planes with images, ray tracing will let the colors come through the transparent areas, but the Z values will not carry through. Therefore, later on down the rendering pipeline some issues may arise. For example, when OverSampling/Aliasing (OSA) is applied, artifacts may ensue as shown. In this example, raytracing is not providing the correct samples for the portion of the ground plane behind the transparent portions of the alpha−mapped tree image plane. To solve this, use ZTransp for the material, not raytracing.

As another alternative, change the texture interpolation filter size to 0.100 or some number smaller than 1.0. This slider is found in the Texture subcontext (F6 Map Image panel.

Raytracing affects the **Color**, but **does not affect the Alpha** channel value that is saved or passed on. Even if the Material is set to Alpha 0, raytracing will treat the material as transparent, but will return an alpha 1 for that channel regardless of the alpha setting inthe material.

ZTransp does affect Alpha channel values in the output. If you are going to be compositing the image later using Alpha over, you probably want to set ZTransp for your material. The disadvantage with ZTransp is that the light can not be bent or refracted.

To set the Alpha value of an object AND use raytracing, set the Pass Index the object (Object and Links panel). Render the image using Ray Transp and the Index pass (Render Layer panel). Then use the Index OB node to mask it out and the Set Alpha node to set the alpha for that portion of the image.

IOR values for Common Materials

The following list provides some Index Of Refraction values to use when RayTraced Transparency is used for varous liquids, solids (gems), and gases:

A		**E**		**J**		**S**	
Acetone	1.36	Ebonite	1.66	Jade, Jadeite	1.64 − 1.667	Sanidine	1.522
Actinolite	1.618	Ekanite	1.600	Jade, Nephrite	1.600 − 1.641	Sapphire	1.757 − 1.779
Agalmatoite	1.550	Elaeolite	1.532	Jadeite	1.665	Sapphire, Star	1.760 − 1.773
Agate	1.544	Emerald	1.560 − 1.605	Jasper	1.540	Scapolite	1.540
Agate	1.540	Emerald Catseye	1.560 − 1.605	Jet	1.660	Scapolite, Yellow	1.555
Air	1.000						
Alcohol	1.329						

Alcohol, Ethyl (grain)	1.36
Alexandrite	1.745
Alexandrite	1.750
Almandine	1.83
Aluminum	1.44
Amber	1.545
Amblygonite	1.611
Amethyst	1.540
Ammolite	1.600
Anatase	2.490
Andalusite	1.640
Anhydrite	1.571
Apatite	1.632
Apophyllite	1.536
Aquamarine	1.575
Aragonite	1.530
Argon	1.000281
Asphalt	1.635
Axenite	1.674 − 1.704
Axinite	1.675
Azurite	1.730
B	
Barite	1.636
Barytocalcite	1.684
Beer	1.345
Benitoite	1.757
Benzene	1.501
Beryl	1.57 − 1.60
Beryl, Red	1.570 − 1.598
Beryllonite	1.553
Brazilianite	1.603
Bromine (liq)	1.661
Bronze	1.18
Brownite	1.567
C	
Calcite	1.486
Calspar	1.486
Cancrinite	1.491
Carbon Dioxide (gas)	1.000449
Carbon Disulfide	1.628
	1.460
Emerald, Synth flux	1.561
Emerald, Synth hydro	1.568
Enstatite	1.663
Epidote	1.733
Ethanol	1.36
Ethyl Alcohol	1.36
Euclase	1.652
F	
Fabulite	2.409
Feldspar, Adventurine	1.532
Feldspar, Albite	1.525
Feldspar, Amazonite	1.525
Feldspar, Labradorite	1.565
Feldspar, Microcline	1.525
Feldspar, Oligoclase	1.539
Flourite	1.434
Formica	1.47
G	
Garnet, Andradite	1.88 − 1.94
Garnet, Demantiod	1.880 − 1.9
Garnet, Demantoid	1.880
Garnet, Grossular	1.738
Garnet, Hessonite	1.745
Garnet, Mandarin	1.790 − 1.8
Garnet, Pyrope	1.73 − 1.76
Garnet, Rhodolite	1.740 − 1.770
Garnet, Rhodolite	1.760
Garnet, Spessartite	1.810
Garnet, Tsavorite	1.739 − 1.744
Garnet, Uvarovite	1.74 − 1.87
K	
Kornerupine	1.665
Kunzite	1.660 − 1.676
Kyanite	1.715
L	
Labradorite	1.560 − 1.572
Lapis Gem	1.500
Lapis Lazuli	1.50 − 1.55
Lazulite	1.615
Lead	2.01
Leucite	1.509
M	
Magnesite	1.515
Malachite	1.655
Meerschaum	1.530
Mercury (liq)	1.62
Methanol	1.329
Milk	1.35
Moldavite	1.500
Moonstone	1.518 − 1.526
Moonstone, Adularia	1.525
Moonstone, Albite	1.535
Morganite	1.585 − 1.594
N	
Natrolite	1.480
Nephrite	1.600
Nitrogen (gas)	1.000297
Nitrogen (liq)	1.2053
Nylon	1.53
O	
Obsidian	1.489
Oil of Wintergreen	1.536
Oil, Clove	1.535
Oil, Lemon	1.481
Oil, Neroli	1.482
Oil, Orange	1.473
Oil, Safflower	1.466
Oil, vegetable (50Â° C)	1.47
Scheelite	1.920
Selenium, Amorphous	2.92
Serpentine	1.560
Shampoo	1.362
Shell	1.530
Silicon	4.24
Sillimanite	1.658
Silver	0.18
Sinhalite	1.699
Smaragdite	1.608
Smithsonite	1.621
Sodalite	1.483
Sodium Chloride	1.544
Spessarite	1.79 − 1.81
Sphalerite	2.368
Sphene	1.885
Spinel	1.712 − 1.717
Spinel, Blue	1.712 − 1.747
Spinel, Red	1.708 − 1.735
Spodumene	1.650
Star Ruby	1.76 − 1.773
Staurolite	1.739
Steatite	1.539
Steel	2.50
Stichtite	1.520
Strontium Titanate	2.410
Styrofoam	1.595
Sugar Solution 30%	1.38
Sugar Solution 80%	1.49
Sulphur	1.960
Synthetic Spinel	1.730
T	
Taaffeite	1.720
Tantalite	2.240
Tanzanite	1.690−1.7

Carbon Tetrachloride	
Carbonated Beverages	1.34 − 1.356
Cassiterite	1.997
Celestite	1.622
Cerussite	1.804
Ceylanite	1.770
Chalcedony	1.544 − 1.553
Chalk	1.510
Chalybite	1.630
Chlorine (gas)	1.000768
Chlorine (liq)	1.385
Chrome Green	2.4
Chrome Red	2.42
Chrome Tourmaline,	1.61 − 1.64
Chrome Yellow	2.31
Chromium	2.97
Chrysoberyl	1.745
Chrysocolla	1.500
Chrysoprase	1.534
Citrine	1.532 − 1.554
Citrine	1.550
Clinohumite	1.625 − 1.675
Clinozoisite	1.724
Cobalt Blue	1.74
Cobalt Green	1.97
Cobalt Violet	1.71
Colemanite	1.586
Copper	1.10
Copper Oxide	2.705
Coral	1.486
Coral	1.486 − 1.658
Cordierite	1.540
Corundum	1.766
Cranberry Juice (25%)	1.351
Crocoite	2.310
Crysoberyl, Catseye	1.746 − 1.755
Crystal	2.000
Cuprite	2.850
Gaylussite	1.517
Glass	1.51714
Glass, Albite	1.4890
Glass, Crown	1.520
Glass, Crown, Zinc	1.517
Glass, Flint, Dense	1.66
Glass, Flint, Heaviest	1.89
Glass, Flint, Heavy	1.65548
Glass, Flint, Lanthanum	1.80
Glass, Flint, Light	1.58038
Glass, Flint, Medium	1.62725
Glycerine	1.473
Gold	0.47
H	
Hambergite	1.559
Hauyn	1.490 − 1.505
Hauynite	1.502
Helium	1.000036
Hematite	2.940
Hemimorphite	1.614
Hiddenite	1.655
Honey, 13% water content	1.504
Honey, 17% water content	1.494
Honey, 21% water content	1.484
Howlite	1.586
Hydrogen (gas)	1.000140
Hydrogen (liq)	1.0974
Hypersthene	1.670
I	
Ice	1.309
Idocrase	1.713
Iodine Crystal	3.34
Iolite	1.522 − 1.578
Iron	1.51
Ivory	1.540
Olivine	1.670
Onyx	1.486
Opal, Black	1.440 − 1.460
Opal, Fire	1.430 − 1.460
Opal, White	1.440 − 1.460
Oregon Sunstone	1.560 − 1.572
Oxygen (gas)	1.000276
Oxygen (liq)	1.221
P	
Padparadja	1.760 − 1.773
Painite	1.787
Pearl	1.530
Periclase	1.740
Peridot	1.635 − 1.690
Peristerite	1.525
Petalite	1.502
Phenakite	1.650
Phosgenite	2.117
Plastic	1.460
Plexiglas	1.50
Polystyrene	1.55
Prase	1.540
Prasiolite	1.540
Prehnite	1.610
Proustite	2.790
Purpurite	1.840
Pyrite	1.810
Pyrope	1.740
Q	
Quartz	1.544 − 1.553
Quartz, Fused	1.45843
R	
Rhodizite	1.690
Rhodochrisite	1.600
Rhodonite	1.735
Rock Salt	1.544
Rubber, Natural	1.5191
Ruby	1.757 − 1.779
Teflon	1.35
Thomsonite	1.530
Tiger eye	1.544
Topaz	1.607 − 1.627
Topaz, Blue	1.610
Topaz, Imperial	1.605 − 1.640
Topaz, Pink	1.620
Topaz, White	1.630
Topaz, Yellow	1.620
Tourmaline	1.603 − 1.655
Tourmaline	1.624
Tourmaline, Blue	1.61 − 1.64
Tourmaline, Catseye	1.61 − 1.64
Tourmaline, Green	1.61 − 1.64
Tourmaline, Paraiba	1.61 − 1.65
Tourmaline, Red	1.61 − 1.64
Tremolite	1.600
Tugtupite	1.496
Turpentine	1.472
Turquoise	1.610
U	
Ulexite	1.490
Uvarovite	1.870
V−W	
Wardite	1.590
Variscite	1.550
Water (0Â° C)	1.33346
Water (100Â° C)	1.31766
Water (20Â° C)	1.33283
Water (gas)	1.000261
Water 35'C (Room temp)	1.33157
Whisky	1.356
Willemite	1.690

D		Rum, White	1.361	Witherite	1.532
Danburite	1.627 − 1.641	Rutile	2.62	Vivianite	1.580
Danburite	1.633			Vodka	1.363
Diamond	2.417			Wulfenite	2.300
Diopside	1.680			**Z**	
Dolomite	1.503			Zincite	2.010
Dumortierite	1.686			Zircon	1.777 − 1.987
				Zircon, High	1.960
				Zircon, Low	1.800
				Zirconia, Cubic	2.173 − 2.21

Previous: Manual/Raytraced Reflections Contents *Next:* Manual/Multiple Materials

User Manual: Contents | Guidelines | Blender Version **2.4**

We normally think of an object as having a single material; a color, maybe with a little texture. This works for modeling many of the simpler objects in the real world. However, the real world is not always that simple, and some objects get very complicated. Blender allows you to have multiple materials for different pieces of a mesh. This section describes how to use the multiple material features of Blender.

To begin, make sure your 3D window is in Solid, Shaded, or Textured mode so that you can see the effects of your changes.

The Material Index

Default Material Index display

The list of materials that are assigned to a mesh is located by viewing the Editing buttons F9 in a Buttons Window on the Link and Materials tab. On that tab, the buttons used for defining and managing multiple materials are boxed in hot pink in the image to the right. These buttons show the active material name (currently blank), a Material Index display (the 0 Mat 0 ? scroller), and some action buttons labeled New, Delete, Select, Deselect, and Assign. The default panel shown to the right shows that the object Suzanne does not have any materials assigned to it, as evidenced by the 0 Mat 0 material index display.

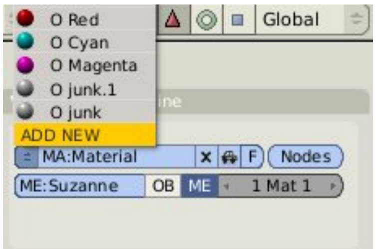
Material Index:

In the < 5 Mat 2 > display, the two small arrows allow you to change the active material − if the object uses more than one of them; the first number (**5** here) gives you the number of materials used by the current object, and the Mat 2 indicates that it's the material of index **2** which is active.

If a mesh does not have any materials (0 Mat 0 is showing), create a quick and simple new material for the object by clicking the New button. A new material will be created and assigned to the object. It's name, "Material" or "Material.00x", will be shown above the Index display. By default, it will be grey. The Index will now display 1 Mat 1

Meshes and Objects

You can re−use the same mesh within Blender many times to create objects. For example, you may create a mesh called "chair" and then create four duplicate objects and put them around a table. Each instance of the mesh, namely each object, may have its own set of materials assigned (one red chair, one blue chair, ...)



Shading Material panel Index display

The **other** place the material index for an object is shown is on the Links and Pipeline panel on the Shading F5 Material Buttons window. The image to the right shows that our Suzanne has one MA:terial called "Material" assigned to it, and that it is the first of one materials assigned to her (the 1 Mat 1 index display). The select/scroll button has been clicked and is showing the list of materials in the blend file, and the ADD NEW option is highlighted. You can use this option to create a new material for the object as previously discussed in preceding sections.

Some of the mesh faces can be assigned to different index. Some of the indexes can use the same material. Changing the material settings (e.g. color) for a material will change the color of all the faces in all the indexes that use that color. To make it even more flexible (and confusing), the same Material can be used by many different meshes in your scene, so changing a material may even change the color of other meshes. In other words, be careful when changing material colors so that you don't have any unwanted effects. If you are ever in doubt, use the Outliner window to see who uses a particular material.

To make different indexes use different colors, you have to Add New materials, one for each of the indexes, as discussed below. There are three steps, and the steps can be done in any order:

- Creating Material Index slots
- Assign Faces to a Material Index
- Associate a Material with a Material Index

Assigning Material Indexes to a Mesh

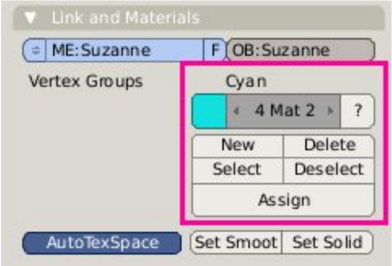
A mesh can have multiple materials assigned to it, and the same material can be re−used many times for many meshes. Blender keeps track of which materials are used with which meshes by keeping a list of materials that a mesh uses. Each material is assigned an index, or slot, on the particular mesh's list.

Adding a Material Index

By clicking the New button, you create a new entry or slot on the object's list. Each time you click the New button, Blender adds the active material to the list.

Active Material

On the Shading Material buttons panels, you are working with a particular material. The material that you are working on is called the Active Material.



Material panel showing 4 indices; index 2 active

So, clicking the New button the first time, when an object that does not have any materials assigned to it, creates a new **material** and assigns it to the first slot. Clicking the New button the second, third, and successive times creates a new **index** and fills that slot with the active material.

The image to the right shows our Editing F9 panel again, this time telling us that Suzanne now has 4 slots, and that we are working with the second, Cyan, as the active material. The color of the material is shown in the swatch.

To select a Material Index, use the left/right scroll arrows (shown in the image as little arrows to the left and right of the 4 Mat 2).

The main, or default material for the object is always in slot 1.

Deleting a Material Index

Take your object out of edit mode. Select the material index slot using the scroll buttons in the material Index display. To delete this index, click the Delete button. Clicking this button disassociates the material with the mesh. It does not delete the material definition from the .blend file, but simply crosses it off the list for that particular object. Any faces of the mesh that may have used that material index will be re−assigned the default material (index/slot 1).

Assigning Face(s) to a Material Index

A "face" is the part of the mesh that actually reflects color and shows the material. A mesh is made up of many faces, connected by edges with corners defined by vertices. All faces of the mesh are assigned to the first material index (x Mat 1) when the mesh was created.

With your object in Edit mode (tab), select the faces that you want to have a particular color (material) in the 3D window. Recall that there are vertice, edge, and face select buttons in the window header. You can select multiple faces using Shift RMB or Border select. Then, in the Material Index panel in the Buttons window, scroll or ensure that the correct index is showing in the x Mat x material index scroller. Press the Assign button to assign those selected faces to the active material index.

Keeping Track

Unlike Vertex Groups, the Material Indexes cannot be named. So, you might want to open a Text Editor window and jot down what Indexes are used for what faces. For example, 1 for base, 2 for ears, 3 for eyes, 4 for nose.

To assign a face to a different index, simply select the face, scroll to the material you want it to show, and press Assign. The face was assigned to a different index, and that assignment is overridden to what you just

specified. If you made a mistake, there's always Undo.

Changing a Material used by a Material Index

When you clicked the New button in the Material Index panel, the active material was assigned to that index. At that moment, the previous index *and* the new index both pointed to and used the *same* material. Since they both used the same material, nothing changed in your display. However, you probably want them to be different.

Use the Shading Material buttons to change the appearance of an active material that is referenced by material indexes. In the Buttons window, the Shading (F5 set), Material buttons, Links and Pipeline panel, use the material index Literal scroller to scroll to the index you want to be a different color. Then use the MA: material scroll/select button to Add New material. All of the faces assigned to that index will now show the new color in your 3D window.

To swap out a material on the list for another, go to the Shading Material buttons, Links and Pipeline panel. Use the MA: material scroll/select button to select a different material for that index (recall the panel also displays the active material in its own x Mat x index display). All of the faces assigned to that index will now have the new color.

Material Users

In the Material select scroll box that shows all the materials in your scene, a number to the right of its name shows the number of objects that share that material. A zero means that the material is unused and readily available without affecting any other objects or faces.

Checking your work using Select, Deselect, Query (?)

To see which faces have which index, in Edit mode, deselect all faces by pressing the A key once or twice. Then click the Select button. In the 3D window, the faces that have the active material will be selected, just as if you selected them manually using RMB.

If you see that a face has the wrong material assigned to it, select it using the RMB, scroll to the correct material index, and press the Assign button.

To see if any faces do NOT have a material assigned to them, use the Deselect button as follows:

- Select all the faces of the object (A once or twice)
- Scroll to the first material index, and click Deselect. All the faces that are assigned that index will be de−selected.
- Repeat the above step for each index.
- Any faces left are unassigned a material index.

To see the material that a face is assigned to, select the face and click the ? question mark query button next to the x Mat x Material Index display. The display will change to show that face's material index.

Some Examples

A soda can has base aluminum, but an image wrapper. The pull tab may be a colored aluminum (three indexes). A chair may have a frame and a cushion (two materials). A old−fashioned pencil has an eraser, metal holder, painted wood stylus, shaved part and graphite (five materials). Sunglasses have a frame and lenses (two materials). The computer I am writing this on has about ten. Our favorite flag (*right*) is one object with three materials (orange, blue and white) The white material index is the default (index 1) and is assigned to both the stars and the background.

Advanced Multiple Materials

A way of mixing materials on a mesh without using geometry for separation is shown in the Mixing Materials with Nodes tutorial.

Previous: Manual/Raytraced Transparency	Contents	*Next:* Manual/Halos

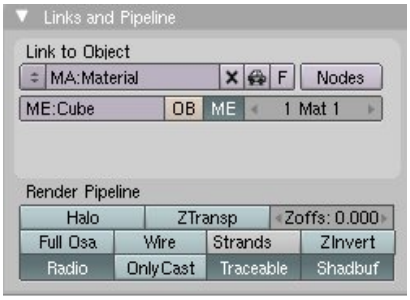
User Manual: Contents | Guidelines | Blender Version **2.45**

Blender provides a set of materials which do not obey the face−shader paradigm and which are applied on a per−vertex rather than on a per−face basis. These are called Halos because you can see them, but they do not have any substance. They are like little clouds of light; although they are not really lights because they do not cast light into the scene like a lamp.

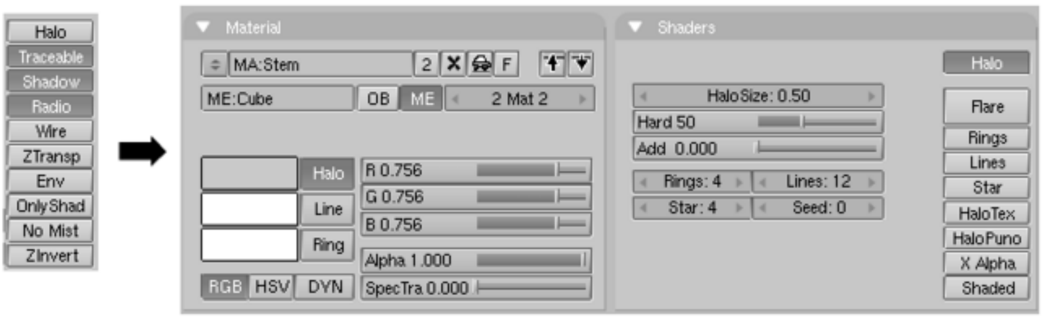
Halo Materials

Halos come in very handy when creating certain special effects, when making an object glow, or when creating a viewable light or fog/atmospherics around an actual light.

Options



Press F5 to display the Material buttons, and then press the Halo button on the Links and Pipeline Panel. The Material and Shaders panels change as shown below in *Halo buttons*.



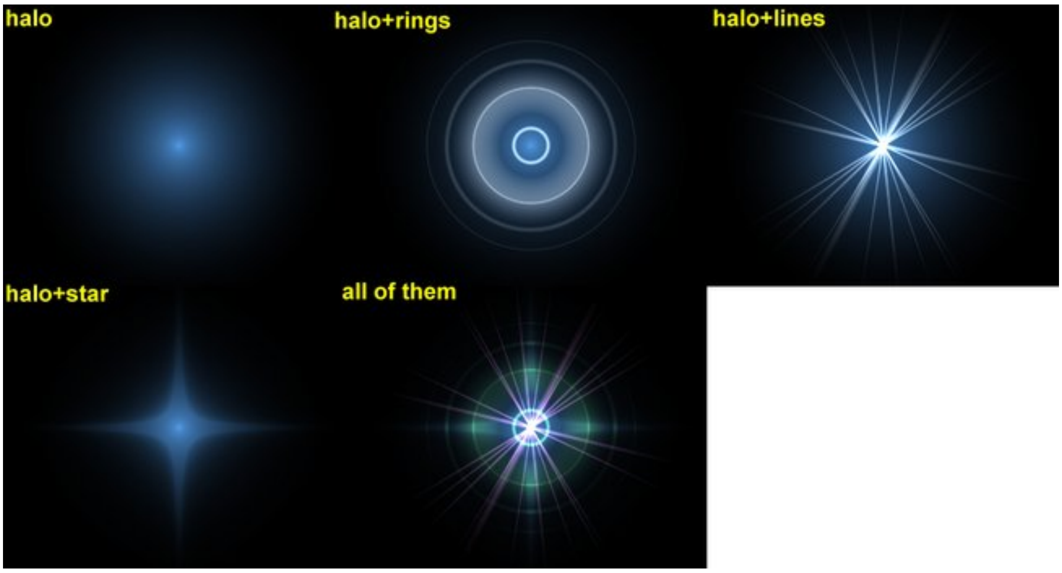
Halo buttons

As you will see in the 3D View, the Mesh faces are no longer rendered. Instead just the vertex is rendered, since that is where each halo will originate. Halos can be hard to find in a crowded scene, so name it well for easy location in the outliner.

In the Material Panel, you now set three colors which, in standard material were Diffuse, Specular and Mirror colors are now relative to three different Halo characteristics:

- Halo: the color of the halo itself,
- Line: the color of any possible line you might want to add
- Ring: the color of any possible ring around the halo

You can not use color ramps. Lines, Rings and an assortment of special effects are available with the relevant toggle buttons, which include Flare, Rings, Lines, Star, Halotex, HaloPuno, X Alpha, and Shaded. *Halo Variations* shows the result of applying a halo material to a single vertex mesh.



Halo Variations

The halo size, hardness and alpha can be adjusted with the pertinent sliders. These are very similar to traditional material settings

The Add slider determine how much the halo colors are 'added to', rather than mixed with, the colors of the objects behind and together with other halos. By increasing Add, the Halo will appear to light up objects that move behind it or through the Halo field.

To set the number of rings, lines, and star points independently, once they are enabled with the relative Toggle Button, use the Num Buttons Rings:, Lines: and Star:. Rings and lines are randomly placed and oriented, to change their pattern you can change the Seed: Num Button which sets the random numbers generator seed.

Halo Texturing

By default, textures are applied to objects with Object coordinates and reflects on the halos by affecting their color, as a whole, on the basis of the color of the vertex originating the halo. To have the texture take effect *within* the halo, and hence to have it with varying colors or transparencies press the HaloTex button; this will map the whole texture to *every* halo. This technique proves very useful when you want to create a realistic

rain effect using particle systems, or similar.

Another Option is Shaded. When shaded is enabled, the Halo will be affect by local light; a lamp will make it brighter and affect its diffuse color and intensity.

Examples

Let's use a halo material to create a dotmatrix display.

- To begin, add a grid with the dimensions 32x16. Then add a camera and adjust your scene so that you have a nice view of the billboard.

- Use a 2D image program to create some red text on a black background, using a simple and bold font (if you are a lazy lizard [I hope this not offensive, I just like how it sounds!], you can just save the picture bellow on your hard driveâ€¦). *Dot matrix image texture.* shows an image 512 pixels wide by 64 pixels high, with some black space at both sides.

Dot matrix image texture.

- Add a material for the billboard, and set it to the type Halo. Set the HaloSize to 0.06 and when you render the scene you should see a grid of white spots.

- Add a Texture, then change to the Texture Buttons and make it an image texture. When you load your picture and render again you should see some red tinted dots in the grid.

- Return to the Material Buttons and adjust the sizeX parameter to about 0.5 then render again; the text should now be centered on the Billboard.

- To remove the white dots, adjust the material color to a dark red and render. You should now have only red dots, but the billboard is still too dark. To fix this enter EditMode for the board and copy all vertices using the Shift D shortcut (take care not to move them!). Then adjust the brightness with the Add value in the MaterialButtons.

Dot Matrix display.

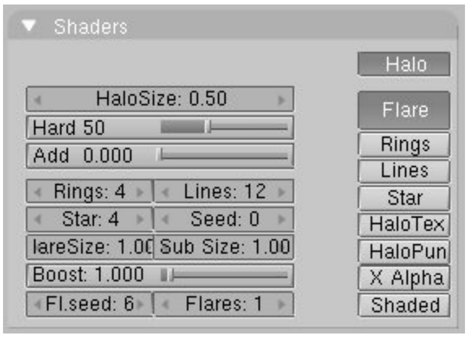
You can now animate the texture to move over the billboard, using the ofsX value in the Texture panel of the

MaterialButtons. (You could use a higher resolution for the grid, but if you do you will have to adjust the size of the halos by shrinking them, or they will overlap. (*Dot Matrix display*).

Note: − Halo materials only work when applied using the first material index. Any material(s) in a subsequent material index will not be rendered.

Lens Flares

Our eyes have been trained to believe that an image is real if it shows artifacts that result from the mechanical process of photography. *Motion blur*, *Depth of Field*, and *lens flares* are just three examples of these artifacts. The first two are discussed in the *chapter_rendering*; the latter can be produced with special halos. A simulated lens flare tells the viewer that the image was created with a camera, which makes the viewer think that it is authentic. We create lens flares in Blender from a mesh object using first the Halo button and then the Flare options in the Shaders Panel of the material settings. Try turning on Rings and Lines, but keep the colors for these settings fairly subtle. Play with the Flares: number and Fl.seed: settings until you arrive at something that is pleasing to the eye. You might need to play with FlareBoost: for a stronger effect (*Lens Flare settings*). (This tool does not simulate the physics of photons travelling through a glass lens; it's just a eye candy.)



Lens Flare settings

Blender's lens flare looks nice in motion, and disappears when another object occludes the flare mesh. (*Lens Flare*).

Lens Flare

Previous: Manual/Multiple Materials	Contents	*Next:* Manual/Node Editor

User Manual: Contents | Guidelines | Blender Version **2.43**

This section explains the window in general, and its header menu options. It also tells you how to enable nodes for use within Blender.

More information on **where to use nodes** please refer to the following pages:

- Nodes for Materials (types of material nodes).
- Nodes for Composition (types of composition nodes).

To learn more about **how to handle nodes themselves** in general please refer to the following page:

- Manual/Using Nodes.

Also there is a reference page available explaining the nodes windows and connected functions:

- Reference/Windows/Nodes.

Accessing The Node Editor

Select the Node Editor window.

First let's enter the node editor by changing our window type to Node Editor. As shown in *Select the Node Editor window*, click on the window type icon and select Node Editor from the popup list. Node maps can get quite large, so use or create a big window. The window has a graph−paper style background and a header.

Each scene within your blend file can have multiple Material Node map and ONE Compositing Node map. The Node Editor window shows either map, depending on the selector position.

Hint

You might want to add a new window layout called 6−Nodes (the list is shown on the User Preferences header at the top of your screen) comprised mostly of one big Node Editor window. My layout has the buttons window at the bottom and a text editor window on the side for me to keep notes. If you have a widescreen display (or even a regular one), you might also want to add a 3D view or UV/Image Editor window to the left side of the Node window layout, so you can work with images or your model while you're manipulating nodes. Having the 3D Preview Render panel open on top of an object is quite useful if you're tweaking material nodes.

By default, the header, when first displayed, is uninitialized as shown:

Default Node Editor header.

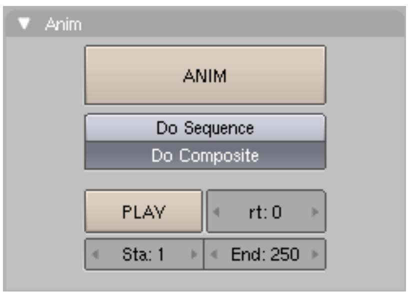
Activating Nodes

- What nodes to use?
  - If you want to work with a material node map, click the ball in the Material/Compositing node set selector. (see *Node Editor Header with Material Nodes enabled.*)
  - If you want to work with a compositing node map, click the face on the Material/Compositing node set selector. (see *Node Editor Header with Compositing Nodes enabled.*)
- To actually activate nodes, click the Use Nodes button.
- The first time that you select either a Material or a Compsiting node map, the Node Editor window will be instantly filled with starter input and output compositing nodes already connected together.

Node Editor Header with Material Nodes enabled.

Node Editor Header with Compositing Nodes enabled.

Required settings for Composition



Use Composition Nodes.

If you are compositing, you must now tell Blender to use the Node map that has been created, and to composite the image using the Node Map. To do so, click on the Do Composite button located below the Animation button. This tells Blender to composite the final image by running it through the composition node map.

From here, you add and connect nodes in a sort of map layout to your heart's content (or physical memory constraints, whichever comes first). But first, let's lay the groundwork by going over the window in general and the header menu options and buttons.

Node Editor Window Actions

When the cursor is in the window, several standard Blender hotkeys and mouse actions are available, including:

Popup menu
: Space − Brings up a main popup menu, allowing you to add, view, select, etc.

Delete
: X or Del − Deletes the selected node(s).

Box select
: B − Starts the bounding box selection process. Position your cursor and LMB click & drag to select a set of nodes.

Cut connections (box)
: LMB click & drag − Starts a box selection, BUT when you let up the mouse button, all threads (connections) within the box are broken.

Undo
: Ctrl Z Very helpful if you forgot to press B before box−selecting, eh?

Redo
: Ctrl Y or Shift Ctrl Z − You can use this if you used "undo" a bit to often :)

Select multiple
: Shift LMB or Shift RMB − Multiple node select.

Grab/Move
: G − Moves your current selection around.

Execute
: E − pumps inputs through the noodle, refreshing everything.

Standard Window Control

Node maps can get pretty hairy (large and complicated, that is). The contents of the window, (the node map) can be panned just like any other Blender window by clicking MMB and dragging about. Wheeling MW up/down or using the keypad
NumPad +/NumPad − will zoom in/out. The window can be resized and combined using the standard window techniques (see *Navigating in 3d Space*).

Node Editor Header

Node Editor Header with Material Nodes enabled.

Node Editor Header with Compositing Nodes enabled.

On the window header, you will see header options:

- View − to see things more clearly;
- Select − to do things more clearly;
- Add − to walk with...err..to add Nodes, organized by type;
- Node − to do things with selected nodes, akin to vertices;
- a Material or Compositing node set selector;
- a Use Nodes button;
- a Free Unused button.

View, Select, and Add Header Menus

These popup menus provide the basic functions:

View
: This menu changes your view of the window, standing in for the standard keyboard shortcuts NumPad + (zoom in), NumPad − (zoom out), HOME (zoom all) or equivalent mouse actions.

Select
: This menu allows you to select a node or groups of nodes, and does the same as typing the hotkey to select all A or start the border select B process.

Add
: This menu allows you to add nodes. Please see the next section for a discussion on the types of nodes that you can add, and what they do. Clicking this menu item is the same as pressing space when the cursor is in the window

Node Header Menu

Show Cyclic Dependencies
: C − Ok, so you've been adding and connecting nodes to your heart's content, and you haven't run out of memory yet. Selecting Show Cyclic Dependencies will show you where you have connected your threads in a circle. For example, you can easily connect a mix output as input to another node, and then connect that node's output back to the mix node input, resulting in a little circle where the image just runs round and round. Left alone, it will eventually get tired and dizzy and crash your computer.

Hide
: H − Hides your selected nodes. Just like vertices in a mesh.

Grouping
: Most importantly, this menu option allows you to create a user−defined group of nodes. This group can then be edited and added to the map. To create a group, select the nodes you want, and then Node â†’ Make Group, or just use the keyboard shortcut Ctrl G. Edit the name using the little input box in the group. Groups are easily identified by their green header and cool names you have picked for them.

Delete
: X − Deletes selected nodes.

Duplicate
: Shift D − Makes an Unlinked copy, with the same settings as the original.

Grab
: G − Moves the little nodes around according to your mouse, just like with meshes.

Duplicate − Faked you out

The new copy is placed **exactly over the old one**. But it isn't the connected one, so playing with the controls will do nothing to your images, even though it **looks** like it's connected with the little threads coming out of the node that is **underneath**. You have to move the duplicated node to reveal the connected node beneath it.

Grab − Reminder Only

Just like my mother−in−law, the menu item does not actually do anything; it's just there to remind you that you can press the G key when your cursor is in the window and actually accomplish something with your life (like rearranging nodes in the window).

Material/Composting Selector Button

Nodes are grouped into two categories, based on what they operate on. Material Nodes operate on a material in use within the blend file. To work with Material Nodes, click on the ball. When you want to work with Compositing nodes, click on the face to show the Compositing Node map.

Use Nodes Header Button

This button tells the render engine to use the node map in computing the material color or rendering the final image, or not. If not, the map is ignored and the basic render of the material tabs or scene is accomplished.

Free Unused Header Button

This button frees up memory space when you have a very complex node map. Recommended.

Previous: Manual/Halos | Contents | *Next:* Manual/Using Nodes

User Manual: Contents | Guidelines | Blender Version **2.43**

Please only proceed after reading Node Editor. You need to be in the Node editor window in order to follow the text below.

Node Basics

What are nodes

"Nodes" (some people call connected nodes "noodles") are individual blocks that perform a certain operation on zero or more inputs or outputs. Some nodes, like the Value or RGB nodes, only output a value. Other nodes, such as the RGB Curves node, take in an image, and output a modified copy. Other nodes, such as the Defocus Node or Vector Blur node, perform even more complex tasks.

Adding and Arranging Nodes

Nodes are added in two ways to the node editor window:

- By clicking the Add menu in the node editor toolbar and picking the type of node you want, or
- By having your cursor in the node editor window and pressing Space and picking a node from the popup Add menu.

In general, try to arrange your nodes within the window such that the image flows from left to right, top to bottom. Move a node by clicking on a benign area and dragging it around. The node can be clicked almost anywhere and dragged about; connections will reshape as a bezier curve as best as possible.

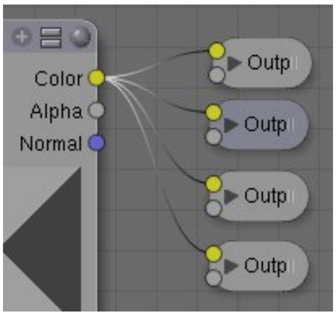
Threads beneath Nodes

Threads (the curves that connect sockets) may reposition behind a node; however they are just that and do not interact with that node in any way.

Sockets

Node Sockets.

Each Node in your node window will have "sockets" (often also referred to as "connectors") which are small colored circles to which input data and output data will be linked (*Node Sockets*).



Node Linking.

There are three colors:

Yellow sockets
: Indicates that color information needs to be input or will be output from the node.

Grey sockets
: Indicates values (numeric) information. It can either be a single numerical value or a so−called "value map" (you can think if a value map as a grayscale−map where the different amount of bright/dark reflects the value for each point.) If a single value is used as an input for a "value map" socket all points of the map are set to this same value.
Common use: Alpha maps and value−options for a node.

Blue/Purple sockets
: Indicates vector/coordinate/normal information.

Between nodes, yellow must be linked to yellow, gray to gray, blue to blue, unless you use a converter which we'll cover later on.

Next to the color in the node you will see the name of that socket. Though not always the case, you can see the name of the socket as what the information is *intended* to be, not necessarily what it *has* to be − for example, I can add a link from an gray socket titled Alpha to the material node's gray Reflection socket and still get a result, they key thing being that it's a gray to gray connection.

There are exceptions where you can mix yellow (e.g. a color−image) and gray (e.g. grayscale) without convertors, Blender normally places a convertor if needed, so feel free to experiment with them. You can use the "Viewer" output nodes as explained in the later sections to see if/how it works.

Connecting and Disconnecting Sockets

You link between sockets by clicking the socket with the LMB and holding to drag the thread to another socket, you then let go once you reach the corresponding socket. To break a link between sockets click the LMB and hold and drag a box around any part (it can be really small) to break the link. From output sockets, multiple threads can be extracted and attached to many nodes (*Node Linking*).

In this case, a copy of each output is routed along a thread. However, only a single thread can be linked to an input socket.

Node Groups

Both material and composite nodes can be grouped. Grouping nodes can simplify the node network layout in

the node editor, making your material or composite 'noodle' (node network) easier to work with. Grouping nodes also creates what are called NodeGroups (inside a .blend file) or NodeTrees (when appending).

If you have created a material using nodes that you would like to use in another .blend file, you can simply append the material from one .blend file to another.

However, what if you would like to create a new material, and use a branch from an existing material node network? You could re−create the branch. Or you could append the material to the new .blend file, then cut and paste the branch that you want into the new material. Both of these options work, but are not very efficient when working across different .blend files.

What if you have created a â€œDepth of Fieldâ€ composite node network and would like to use it in another .blend file? Here again, you could re−create the network, but this is not very efficient.

A better method of re−use for either material node branches or composite node networks would be to create groups of nodes. These groups will then be made available through the Blender's library and standard appending method.

Grouping Nodes

To create a node group, in the node editor, select the nodes you want to include, then press Ctrl G or Space â†' node â†' make group. A node group will have a green title bar. All of the selected nodes will now be minimized and contained within the group node. Default naming for the node groups is NodeGroup, NodeGroup.001 etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful. When appending node groups from one .blend file to another, Blender does not make a distinction between material node groups or composite node groups, so I recommend some naming convention that will allow you to easily distinguish between the two types. For example, name your material node branches Mat_XXX, and your composite node networks Cmp_XXX.

What NOT to include in your groups.

> Material node groups should not include material nodes or output nodes. If you include a material node in your group, you will end up having the material node appear twice, once inside the group, and once outside the group in the new material node network. If you include an output node in the group, there will not be an output socket available from the group.
>
> Composite node groups can not include a Render Layer node (Blender wont let you), and should not contain a Composite output node. Here again, if they include any connected Output node (Viewer, Split Viewer, etc) then the Group will not have an image output socket.

Editing Node Groups

With a group node selected, pressing Tab expands the node to a window frame, and the individual nodes within it are shown to you. You can move them around, play with their individual controls, re−thread them internally, etc. just like you can if they were a normal part of your editor window. You will not be able to thread them to an outside node directly from them; you have to use the external sockets on the side of the Group node. To add or remove nodes from the group, you you need to ungroup them.

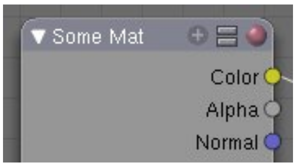
Ungrouping Nodes

The Alt G command destroys the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread internal nodes to other nodes in your workspace.

Appending Node Groups

Once you have appended a NodeTree to your .blend file, you can make use of it in the node editor by pressing Space â†’ Add â†’ Groups, then select the appended group. The "control panel" of the Group is the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

Node Controls



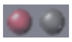
Top of a Node.

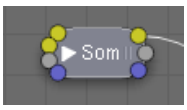
At the top of a node there are up to 4 visual controls for the node (*Top of a Node*). Clicking these controls influences how much information the node shows.

 ***Arrow***
The arrow on the left collapses the node entirely (*Collapsing Arrow*).

 ***Plus sign (+)***
The "plus" icon collapses all sockets that do not have a thread connected to it (*Plus Sign*).

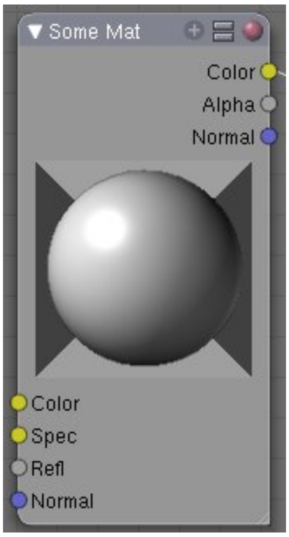
 ***Two squares (=)*** *or "Equal sign"*
The icon with the two squares collapse all the items in a node that have boxes with information in them (*Menu Collapse*).

 ***Sphere***
The sphere icon collapses the viewing window (if the node has one) (*Sphere*).
If the Sphere is **red** this can have 3 reasons:
◊ It's the only effective output Composite node in the compositor.
◊ It's the only effective material Output node (the first one that is added).
◊ If it's a Material input node that has a Material (MA:) assigned to it.



Collapsing Arrow.

Plus Sign.

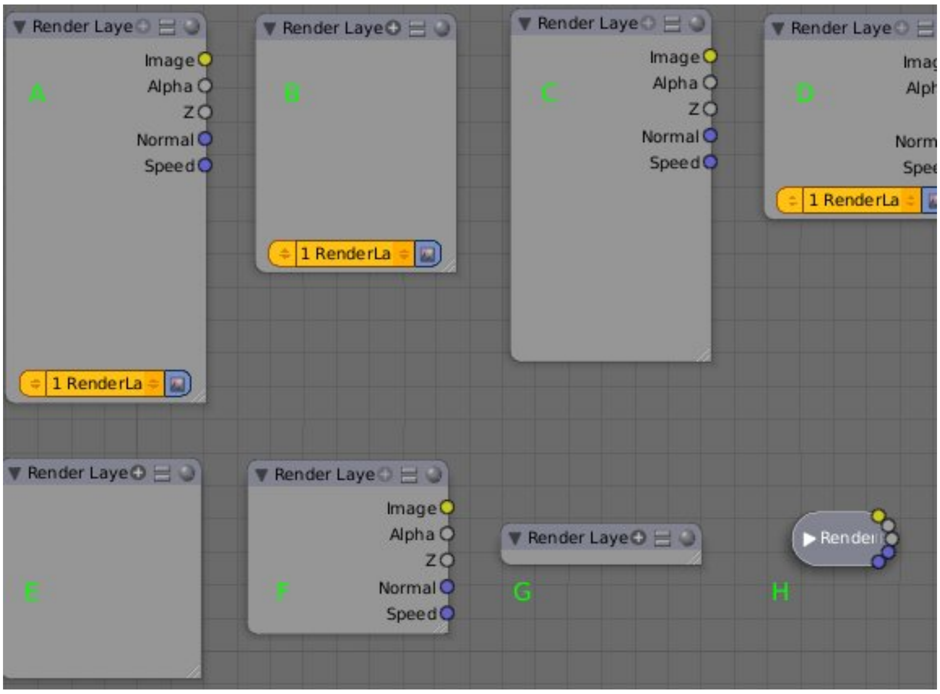


Menu Collapse.

Sphere.

In Combination.

The later three can be used in varying combinations with each other. The arrow that collapses the entire node can only be used in combination with the plus sign (*In Combination*).



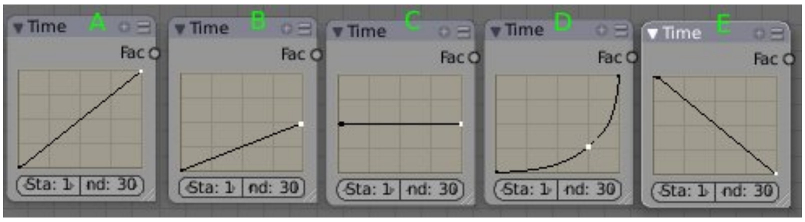
Top sizing controls of a Node
A) Normal, **B**) + Sign clicked, **C**) = Sign clicked, **D**) Sphere clicked, **E**) + and = clicked, **F**) = and Sphere clicked, **G**) All three clicked **H**) Arrow clicked.

Node Sizing

Fine Sizing of an individual node can also be accomplished somewhat by clicking LMB and dragging in the lower right−hand corner (where the little slanted lines are).

Node Curves

Some nodes have a curve area that translates an input value to an output value. You can modify this curve shape by clicking on a control point and moving it, or adding a control point. Some examples are shown below:



Modifying a curve node.

Every curve starts out as a straight line with a slope of 1. (My daughter NEVER thought she would use her high school algebra. Ha!) The curve starts out with two tiny black control points at each end of the line. Clicking LMB on a control point selects it and it turns white.

Changing the curve affects how the output is generated. The input, X, usually proceeds linearly (at regular intervals) across the bottom axis. Go up until you hit the curve, and then over to the right to determine the Y output for that corresponding X. So, for the second example, as X goes from 0 to 1.0 across the bottom, Y varies from 0.0 to 0.5. In the third, as X goes from 0.0 to 1.0 across the bottom, Y stays constant at 0.5. So, in the picture above, these curves have the following affect on time: **A** don't affect, **B** slow down, **C** stop, **D** accelerate, and **E** reverse time.

The "Curves" widget is a built−in feature in Blender's UI, and can be used anywhere, provided the curve data itself is being delivered to this widget. Currently it is in use in the Node Editor and in the UV Window.

This widget will map an input value horizontally and return the new value as indicated by the height of the curve.

RGB Curves

Multiple curves can be edited in a single widget. The typical use, RGB curves, has "Combined" result or "Color" ("C") as the first curve, and provides curves for the individual R, G, and B components. All four curves are active together, the "C" curve gets evaluated first.

Selecting curve points

- LMB always selects 1 point and deselects the rest.
- Hold Shift while clicking to extend the selection or select fewer points.

Editing curves

- LMB click&drag on a point will move points.
- A LMB click on a curve will add a new point.
- Dragging a point exactly on top of another will merge them.
- Holding Shift while dragging snaps to grid units.
- Ctrl LMB adds a point.
- Use the X icon to remove selected points.

Editing the view

The default view is locked to a 0.0−1.0 area. If clipping is set, which is default, you cannot zoom out or drag the view. Disable clipping with the icon resembling a #.

- LMB click&drag outside of curve moves the view
- Use the + and − icons to zoom in or out.

Special tools

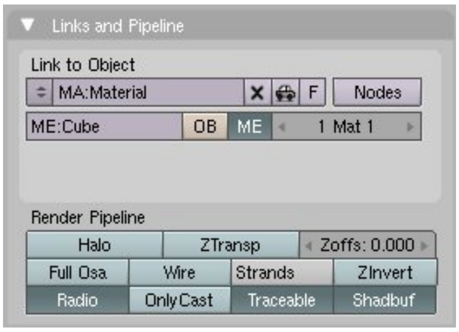
The wrench icon gives a menu with choices to reset a view, to define interpolation of points, or to reset the curve.

Previous: Manual/Node Editor Contents *Next:* Manual/Node Materials

User Manual: Contents | Guidelines | Blender Version **2.42**

Introduction

In addition to creating materials as just described using all the settings on all the materials panels, Blender allows you to create a material by routing basic materials through a set of nodes. Each node performs some operation on the material, changing how it will appear when applied to the mesh, and passes it on to the next node. In this way, very complex material appearances can be achieved.



Links and Pipeline tab

You should already be familiar with general material concepts and how to create materials/textures using the material panel. You should also have a general understanding of the texture coordinate systems available in Blender (e.g. Orco, UV, etc.). Also, when reading this I intend to purposely skip aspects of a node because in later sections you will see the function expanded upon. Each section builds off the previous.

I will begin by saying that the node system does not make the material pane obsolete. Many features and material settings are still only accessible through the material panel (e.g. Ray Mirror). However with the advent of nodes, more complex and fantastic materials can be created since we now have greater control. So letâ€™s begin with a normal material (*Links and Pipeline tab*).

Here we have the standard material we have added to a cube mesh. I could, as I have in the past, add color and other settings to this material and it would certainly look nice. But letâ€™s say I am just not getting what I am looking for? What if I what to control the creation more tightly or add more complexity? Here is where nodes come in.

Making this node map is accomplished by working in a Node Editor window. This section covers:

- Node editor window and basic controls
- How to work with a node (general)
- The specific types of nodes available for materials

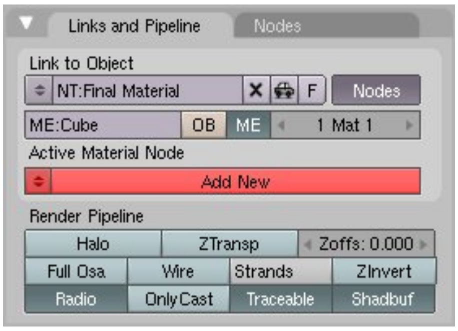
Accessing The Node Editor

Select the Node Editor window

First lets enter the [node editor](#) (*Select the Node Editor window*) and make sure that the node editor has the material node button (the sphere icon) pressed (*Node Editor Toolbar*), not the composite node button.

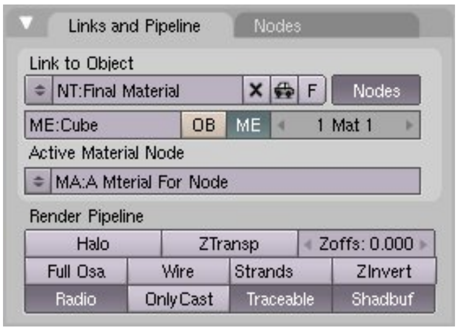
Node Editor Toolbar for Materials

Enabling Node Materials in the Material Buttons



Links and Pipeline Node tab

Letâ€™s take the base material (*Links and Pipeline tab*) and hit the Nodes button next to the material name in the material panel or the node editor. You will see a change in the material panel (*Links and Pipeline Node tab*).



New Node Tree Material

What you have just done is told Blender to make the material you were on (in this case "Final Material") to become the node tree (hence the new name − NT: versus MA:). Under the node tree you can see that it is asking you to add a new material (*Links and Pipeline Node tab*). Once you do (*New Node Tree Material*) you will create a material (MA:) that is under node tree. After adding a material to the node tree, two nodes will appear in the node editor − a material node, and an output node.

It is important to note that you can add a new material (which you can edit and change like any other material in the material panel), add an already created material or append a material from another blender file, and also use the material that you used to create the node tree.

External Links

- Material Nodes Overview − Explains the Material node editor, material node types and some examples of node setups and node−groups.
- Blender Material Nodes − Changelog for the Blender version that introduced material nodes.

Previous: Manual/Using Nodes	Contents	*Next:* Manual/Material Nodes

User Manual: Contents | Guidelines | Blender Version **2.42**

Please be sure to read **Manual/Node Materials** before reading this page.

The material nodes allow you to manipulate a material by routing it through a map of connected nodes. A starting material is routed through different nodes that do various things to the material, combined with other inputs or put back together, and finally output where it can be applied to a mesh, halo, particle, etc.

Materials can be split into their RGB components, combined (mixed) with other inputs, and layered on top of one another. In the real world, this process is accomplished by mixing specific paints, thinners, surface prep, and then painting using various techniques under environmental conditions.

This section is organized by type of node, which are grouped based on similar functions:

- **Input** − Introduces a material or component to the node map.
- **Output** − Displays the result in progress as a small image.
- **Color** − Manipulates the colors of the material.
- **Vector** − Change the way light is reflected off the material.
- **Convertors** − Convert colors to other material colors.
- **Groups** − User−defined groups of nodes.
- **Dynamic** − Custom nodes defined by Python. These are also known as PyNodes.

The simplest way to add a node is to put your cursor in a Node Editor window, and then press Space and click on Add. The popup menu will extend to show you these types of nodes. Click on a node type and the popup menu will extend again to show you specifc node types available.

Material Input Nodes

A starting material is created in the Materials Panel. The Nodes button is enabled to add that material to the list of noded materials shown in the Node Editor window header. Other inputs to the node map include:

- A value
- A color
- A texture
- Geometry

Material Node

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Input â†’ Material

Material node

The Material node is used to add a material to the node program. Materials can be anything from pure shading to fully layered with textures. It inputs the main attributes of a material (color, alpha and normal vector) into the map.

Output

Materials can output color (which includes shading and any textures assigned to it), alpha, and the final normal calculated from any textures it has.

- Color
- Alpha
- Normal

Input

Materials can take inputs for colors, inputs for diffuse color and specularity color, a value for reflectivity, and a normal.

- Color − The base color of the paint. Can be set
    - manually by LMB clicking on the color swatch applet next to the socket, choosing a color using the control panel that pops up, and pressing Enter
    - based on an Active Material which is specified using the material panels, or
    - plugged in from an RGB color generator.
- Spec − The color that is reflected as you get perpendicular to the light source reflecting off the surface. The color can be
    - plugged in from another node or
    - set manually by LMB clicking on and using the color swatch applet.
- Refl: − The degree to which the material reflects light and gives off its color. The value can be provided by another node or set manually.
- Normal − The lighting condition.

Controls

MA:Material field
: You can browse and select materials here.

Diff toggle
: Turn on/off Diffuse Color.

Spec toggle
: Turns on/off Specularity calculation.

Neg toggle
: Inverts the material input normal when activated (which, of course, is a combination of the 3D normal given to it by the 3D object plus the normal input point).

Normal Override

The normal input socket does not in any way blend the source normal with the underlying geometry. Any plugged in Geometry here overrides the Normal lighting conditions.

Using the Material Node with Specularity

Material Node using Specularity

To make a material node actually generate a color, you have to specify at least a basic input color, and optionally a specularity color. The specularity color is the color that shines under intense light.

For example, consider the mini−map to the right. The base color, a dark blue, is connected from an RGB color generator node to the Color input socket. The specular color, yellow, is connected to the Spec input. Under Normal lighting conditions on a flat surface, this material will produce a deep blue color and, as you approach a spot perpendicular to the light, you will see the yellow specular color mix in.

Enable Spec

To see specularity, you have to enable it by clicking the blue Spec button located just below the material color swatch in the node.

Value Node

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Input â†’ Value

Value Node

The Value node has no inputs; it just outputs a numerical value (floating point spanning 0.00 to 1.00) currently entered in the NumButton displayed in its controls selection.

Use this node to supply a constant, fixed value to other nodes' value or factor input sockets.

RGB Node

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Input â†’ RGB

RGB Node

The RGB node has no inputs. It just outputs the Color currently selected in its controls section; a sample of it is shown in the top box. In the example to the right, a gray color with a tinge of red is slected.

To change the brightness and saturation of the color, LMB click anywhere within the square gradient. The current saturation is shown as a little circle within the gradient. To change the color itself, click anwhere along the rainbow Color Ramp.

Texture Node

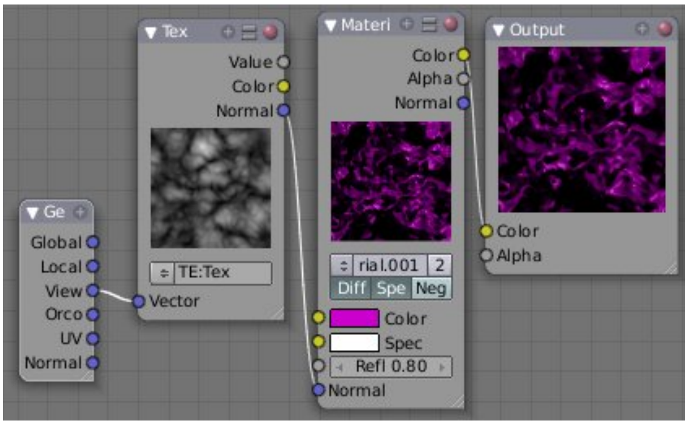
Panel: Node Editor Material Nodes

Menu: Shift A â†’ Input â†’ Texture

A texture, from the list of textures available in the current blend file, is selected and introduced through the value and/or color socket.

Library

Please read up on the Blender Library system for help on importing and linking to textures in other blender files



In the example to the right, a cloud texture, as it would appear to a viewer, is added to a base purple material, giving a velvet effect.
Note that you can have multiple texture input nodes. In the (old) panel way, multiple textures were assigned to channels and each channel was checked on or off to be applied to the material. With nodes, you simply add the textures to the map and plug them into the map.

Geometry Node

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Input â†’ Geometry

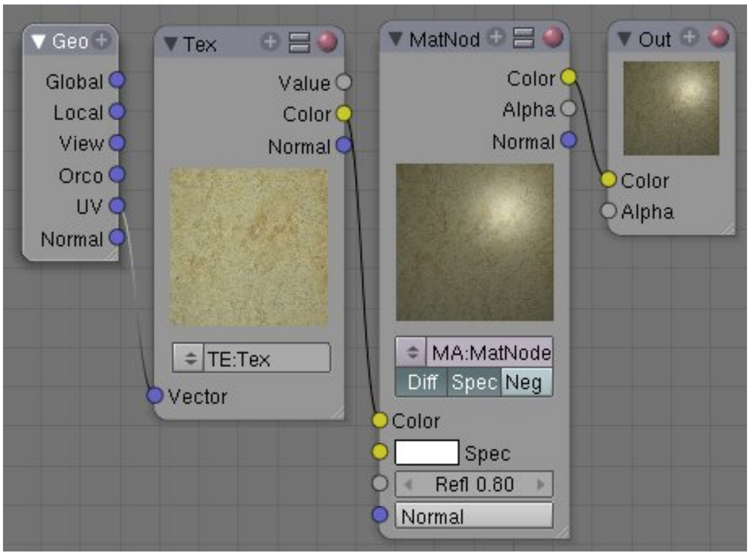
Geometry node

The geometry node is used to specify how light reflects off the surface. This node is used to change a material's Normal response to lighting conditions.

Note

These are exactly the same settings as in the Map Input panel for Textures, though a few settings − like Stress or Tangent − are missing here. Normally you would use this node as input for a Texture Node.

Geometry Node Example using a UV image



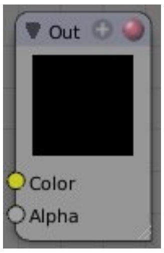
Setup to render an UV−Mapped Image Texture.

E.g.: To render an UV−mapped image, you would use the UV output and plug it into the Vector Input of a texture node. Then you plug the color output of the texture node into the color input of the material node − which corresponds to the setting on the Map To panel.

Material Output Nodes

Panel: [Node Editor](#) [Material Nodes](#)

Menu: Shift A â†’ [Output](#) â†’ Output



Output node

At any point, you may want to see the work in progress, especially right after some operation by a node. Simply create another thread from the output socket of the node to the picture input socket of an Output node to see a mini−picture.

Connect the alpha channel to set/see transparency.

Effective Output Node

The only Output node that is used for the Material in the end (i.e the only non−Preview) has a little **red sphere** on the upper right.

Material Color Nodes

These nodes play with the colors in the material. The choices are:

- Mix
- RGB Curves

Mix Node

Panel: Node Editor â†' Material Nodes

Menu: Shift A â†' Color â†' Mix

This node mixes a base color or image (threaded to the top socket) together with a second color or image (bottom socket) by working on the individual and corresponding pixels in the two images or surfaces. The way the output image is produced is selected in the drop−down menu. The size (output resolution) of the image produced by the mix node is the size of the base image. The alpha and Z channels (for compositing nodes) are mixed as well.

Not one, not two, but count 'em, *sixteen* mixing choices include:

Mix
: The background pixel is covered by the foreground using alpha values.

Add
: The pixels are added together. *Fac* controls how much of the second socket to add in. Gives a bright result.
The "opposite" to Subtract mode.

Subtract
: The foreground pixel (bottom socket) is subtracted from the background one. Gives a dark result.
The "opposite" to Add mode.

Multiply
: Returns a darker result than either pixel in most cases (except one of them equals white=**1.0**). Completely white layers do not change the background at all. Completely black layers give a black result.
The "opposite" to Screen mode.

Screen
: Both pixel values are inverted, multiplied by each other, the result is inverted again. This returns a brighter result than both input pixels in most cases (except one of them equals **0.0**). Completely black layers do not change the background at all (and vice versa) − completely white layers give a white result.

The "opposite" of Multiply mode.

Overlay
A combination of Screen and Multiply mode, depending on the base color.

Divide
The background pixel (top socket) is divided by the second one: if this one is white (= **1.0**), the first one isn't changed; the darker the second one, the brighter is the result (division by **0.5** − median gray − is same as multiplication by **2.0**); if the second is black (= **0.0**, zero−division is impossible!), Blender doesn't modify the background pixel.

Difference
Both pixels are subtracted from one another, the absolute value is taken. So the result shows the distance between both parameters, black stands for equal colors, white for opposite colors (one is black, the other white). The result looks a bit strange in many cases. This mode can be used to invert parts of the base image, and to compare two images (results in black if they are equal).

Darken
Both pixels are compared to each other, the smaller one is taken. Completely white layers do not change the background at all, and completely black layers give a black result.

Lighten
Both parameters are compared to each other, the larger one is taken. Completely black layers do not change the image at all and white layers give a white result.

Dodge
Some kind of inverted Multiply mode (the multiplication is replaced by a division of the "inverse"). Results in lighter areas of the image.

Burn
Some kind of inverted Screen mode (the multiplication is replaced by a division of the "inverse"). Results in darker images, since the image is *burned* onto the paper, er..image (showing my age).

Color
Adds a color to a pixel, tinting the overall whole with the color. Use this to increase the tint of an image.

Value
The RGB values of both pixels are converted to HSV values. The values of both pixels are blended, and the hue and saturation of the base image is combined with the blended value and converted back to RGB.

Saturation
The RGB values of both pixels are converted to HSV values. The saturation of both pixels are blended, and the hue and value of the base image is combined with the blended saturation and converted back to RGB.

Hue
The RGB values of both pixels are converted to HSV values. The hue of both pixels are blended, and the value and saturation of the base image is combined with the blended hue and converted back to RGB.

Color Channels

There are (at least!) two ways to express the channels that are combined to result in a color: RGB or HSV. RGB stands for the Red,Green,Blue pixel format, and HSV stands for Hue,Saturation,Value pixel format.

A
: Click the green Alpha button to make the mix node use the Alpha (transparency) values of the second (bottom) node. If enabled, the resulting image will have an Alpha channel that reflects both images' channels. Otherwise, (when not enabled, light green) the output image will have the Alpha channel of the base (top input socket) image.

Fac
: The amount of mixing of the bottom socket is selected by the Factor input field (Fac:). A factor of zero does not use the bottom socket, whereas a value of **1.0** makes full use. In Mix mode, **50:50** (**0.50**) is an even mix between the two, but in Add mode, **0.50** means that only half of the second socket's influence will be applied.

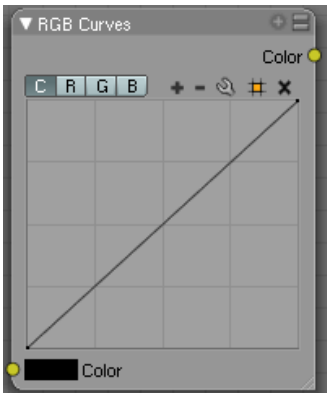
Using Dodge and Burn (History Lesson)

Use the dodge and burn mix methods in combination with a mask to affect only certain areas of the image. In the old darkroom days, when yes, I actually spent hours in a small stinky room bathed in soft red light, I used a circle cut out taped to a straw to dodge areas of the photo as the exposure was made, casting a shadow on the plate and thus limiting the light to a certain area. To do the opposite, I would burn in an image by holding a mask over the image. The mask had a hole in it, letting light through and thus 'burning' in the image onto the paper. The same equivalent can be used here by mixing an alpha mask image with your image using a dodge mixer to lighten an area of your photo. Remember that black is zero (no) effect, and white is one (full) effect. And by the way, ya grew to like the smell of the fixer, and with a little soft music in the background and the sound of the running water, it was very relaxing. I kinda miss those dayz.

RBG Curves Node

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Color â†’ RGB Curves



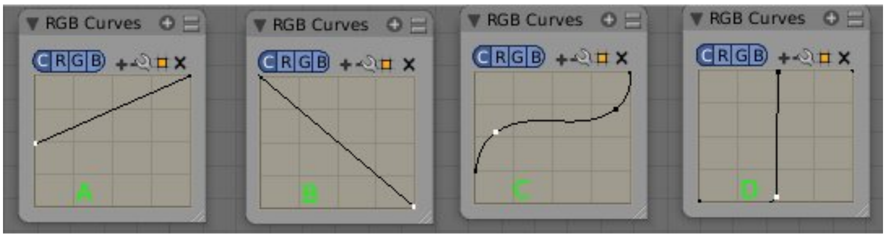
RGB Curves node

For each color component channel (RGB) or the composite (C), this node allows you to define a bezier curve that varies the input (across the bottom, or x−axis) to produce an output value (the y−axis). By default, it is a straight line with a constant slope, so that .5 along the x−axis results in a .5 y−axis output. Click and drag

along the curve to create a control point and to change the curve's shape. Use the X to delete the selected (white) point.

Clicking on each C R G B component displays the curve for that channel. For example, making the composite curve flatter (by clicking and dragging the left−hand point of the curve up) means that a little amount of color will result in a lot more color (a higher Y value). Effectively, this bolsters the faint details while reducing overall contrast. You can also set a curve just for the red, and for example, set the curve so that a little red does not show at all, but a lot of red does.

Here are some common curves you can use to achieve desired effects:



A) Lighten **B**) Negative **C**) Decrease Contrast **D**) Posterize

Material Vector Nodes

Mapping Node

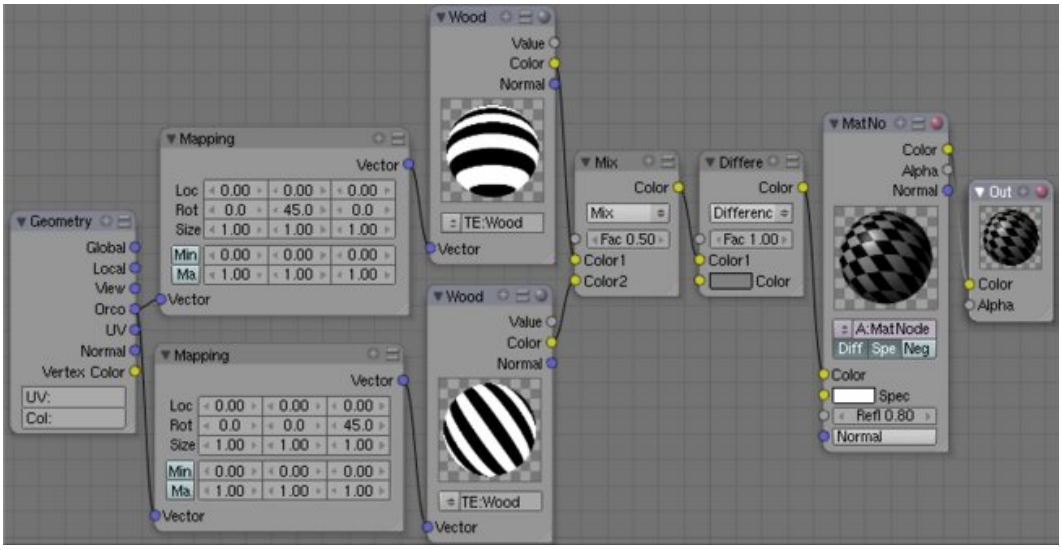
Panel: [Node Editor] â†’ [Material Nodes]

Menu: Shift A â†’ [Vector] â†’ Mapping

Essentially mapping node allows the user to modify a mapping. It is possible to use it do same operations as the Map Input panel found in Material buttons allows. It also makes it possible to do several things that are not possible in Map Input. Mapping can be rotated and clamped if desired. Currently mapping node supports only flat mapping type though.

The controls of the node have been ordered in X, Y, Z order. If you want to use the clamping options, try enabling Min and Max.

Mapping Node Example



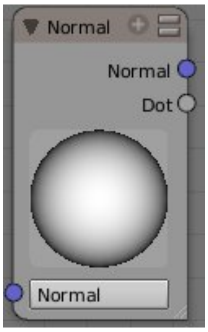
Using mapping nodes to produce amazing chess checkers texture.

This simple example shows one possible way to use mapping nodes to produce interesting textures. As you can see simply by mapping same texture a bit differently can make it useful. Using the same technique it would be easy to mimic different kind of stripy fabrics.

Normal Node

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Vector â†’ Normal



Normal node

The Normal node generates a normal vector and a dot product. Click and Drag on the sphere to set the direction of the normal.

This node can be used to input a new normal vector into the mix. For example, use this node as an input to a Color Mix node. Use an Image input as the other input to the Mixer. The resulting colorized output can be easily varied by moving the light source (click and dragging the sphere).
The (face) normal is the direction of the face in relation to the camera. You can use it to do the following:

- Use this node to create a fixed direction –> output Normal.

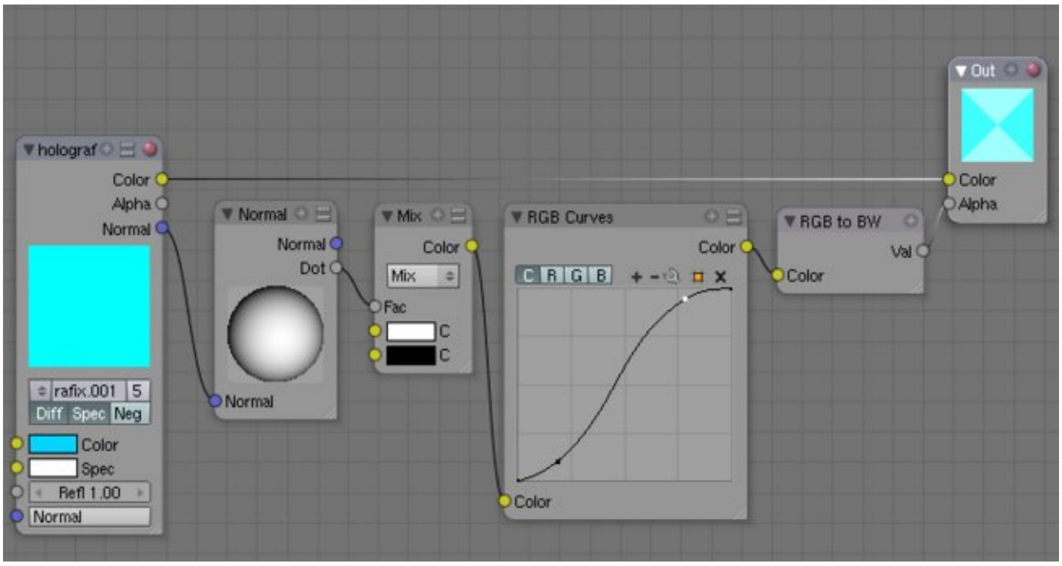
- Calcuate the Dot−Product with the Normal−Input. The Dot−Product is a scalar value (a number).
    - If two normals are pointing in the same direction the Dot−Product is 1.
    - If they are perpendicular the Dot−Product is zero (0).
    - If they are antiparallel (facing directly away from each other) the Dot−Product is −1. *And you never thought you would use the Vector Calculus class you took in college − shame on you!*

So now we can do all sorts of things that depends on the viewing angle (like electron scanning microscope effect or some of techniques described in the section Tutorials/Textures/Map Input Techniques). And the best thing about it is that you can manipulate the direction interactively.

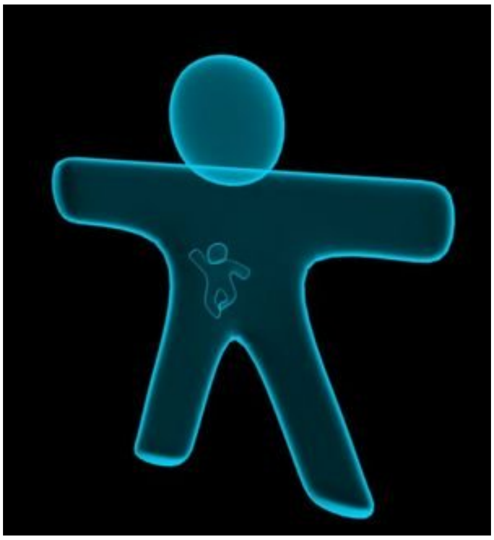
One caveat

The normal is evaluated per face, not per pixel. So you need enough faces, or else you don't get a smooth result

Normal Node Example



Using the *Dot*−Product for viewing angle dependent material, in this case the *Alpha*−Value.



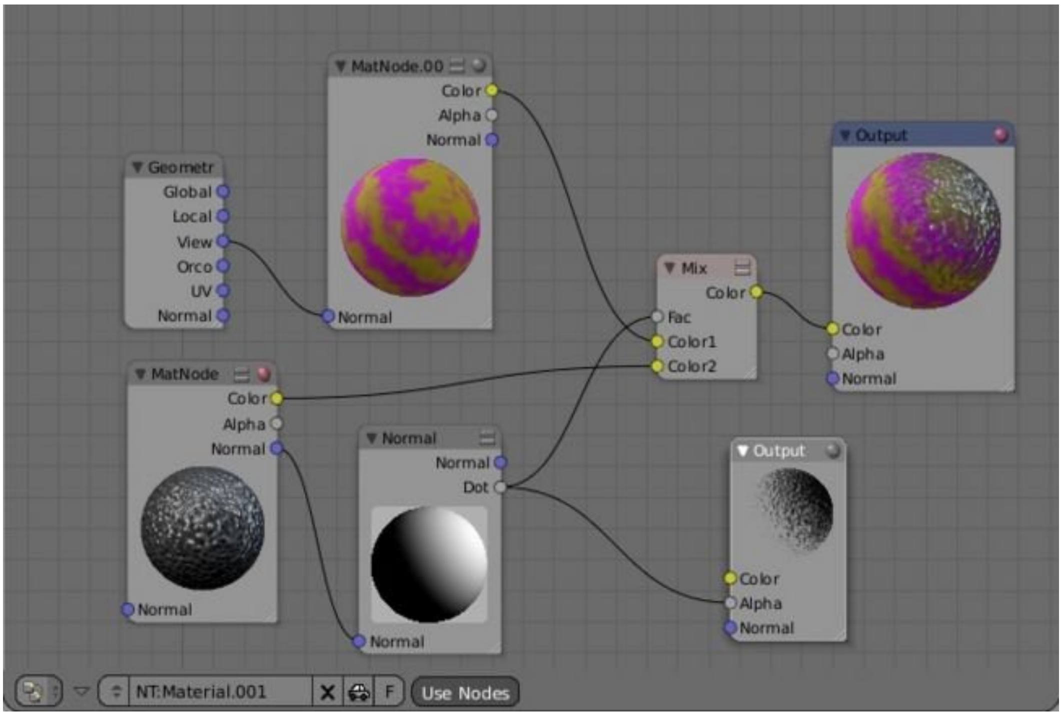
What a surprise − Gus is pregnant!

In the shown example the Dot−Product is used to govern the Alpha−Value of the material. The RGB Curves node is used to sharpen the blend between black and white (which results in a range from fully transparent to fully opaque).

The material is an ordinary blue/cyan material with a high Emit value and Z−Transp activated; the background is black. So, as the face angle gets closer to pointing directly at the camera, the material gets more transparent. As the face gets closer to pointing at a right angle to the camera (in this case facing up or down), it gets more opaque.

Using Normal to reflect bumpy−looking textures

We can use the Normal node to shift the reflection, and thus the shinyness, of a material as shown below. This effect can also be done without nodes, using the materials and texture panels. However, using Nodes allows you to graphically see what is going on to create the final material. The result is that, even though the surface mesh in your model is physically smooth, the reflection makes it look like it has a very fine bumpyness or uneven coating to it, like a really bad paint job or surface prep job was done. This is key to making realistic−looking surfaces.



Using Normal for bump effect

This map starts out using the geometry node to modify a base material called MatNode.001, shown in the header as MatNode.00 (the "1" was cut off to make the Node skinny, which may happen to you). The Geometry node puts out the View vector set, which flattens the color (removes any shine). MatNode.001 is itself a base tan color with a marble texture that affects and mixes in a purple where the texture is black.

A second material, MatNode, which is the default gray color with a noise texture, is routed through our handy Normal node. The ball has been rolled up and to the right. That puts out a mask that you can see in the Output

viewer that is showing the Dot product output. Mixing that mask with the marbleized color and noise material/texture creates the final output for Material.001. As an exercise for the reader, giving the MatNode a color (instead of the grey) would create a specularity effect.

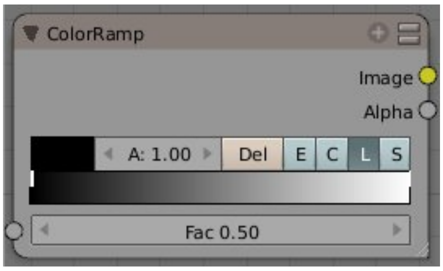
Material Convertor Nodes

As the name implies, these nodes convert the colors in the material in some way.

ColorRamp Node

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Convertors â†’ ColorRamp



ColorRamp node

The ColorRamp Node is used for mapping values to colors with the use of a gradient. It works exactly the same way as a colorband for textures and materials, using the Factor value as a slider or index to the color ramp shown, and outputting a color value and an alpha value from the output sockets.

By default, the ColorRamp is added to the node map with two colors at opposite ends of the spectrum. A completely black black is on the left (Black as shown in the swatch with an Alpha value of 1.00) and a whitewash white is on the right. To select a color, LMB click on the thin vertical line/band within the colorband. The example picture shows the black color selected, as it is highlighted white. The settings for the color are shown above the colorband as (left to right): color swatch, Alpha setting, and interpolation type.

To change the hue of the selected color in the colorband, LMB click on the swatch, and use the popup color picker control to select a new color. Press enter to set that color.

To add colors, hold Ctrl down and Ctrl LMB click inside the gradient. Edit colors by clicking on the rectangular color swatch, which pops up a color−editing dialog. Drag the gray slider to edit A:lpha values. Note that you can use textures for masks (or to simulate the old "Emit" functionality) by connecting the alpha output to the factor input of an RGB mixer.

To delete a color from the colorband, select it and press the Delete button.

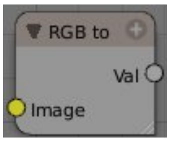
When using multiple colors, you can control how they transition from one to another through an interpolation mixer. Use the interpoloation buttons to control how the colors should band together: Ease, Cardinal, Linear, or Spline.

Use the A: button to define the Alpha value of the selected color for each color in the range.

RGB to BW Node

Panel: Node Editor Material Nodes

Menu: Shift A â†' Convertors â†' RGB to BW



RGB to BW node

This node converts a color image to black−and−white.

Connecting Output Socket

When you connect the output Val socket to an input socket that excepts an Image, Blender automatically inserts a ColorRamp node, to translate the output value to a material color.

Groups of Nodes

Panel: Node Editor Material Nodes

Menu: Shift A â†' Groups

Node maps can get quite complex. Blender allows you to group a set of nodes together to both save space and help you conceptualize what the net effect of a mini−map does to a material/image. This menu selection shows the names of the groups of nodes that you have defined. Select any one to add that group to the map.

Create a Node Group

A group is created by Shift−clicking all the nodes you want in the group, and then selecting Node −> Make Group (Ctrl G). The group node is shown with a green bar, and the name is shown in an editable field (Shift−click on the name to enter EditMode and change the name to something that represents what that group does. The input sockets to the group is the input sockets of contained nodes, and akin for output(s).

Editing Node Groups

You can select the group and press Tab (or select Node −> Edit Group) to enter/leave the group.

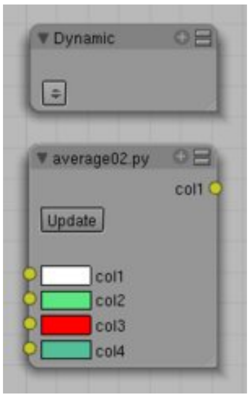
Ungroup Nodes

You can select the group and press Alt G (or select Node −> Ungroup) to convert the grouped nodes back to normal.

Dynamic Nodes (PyNodes)

Panel: Node Editor Material Nodes

Menu: Shift A â†’ Dynamic



PyNodes. Above one is in default state. One below has a script loaded into it.

Dynamic nodes allow you to write your own custom nodes. These nodes are written in Python. See API for more specific information.

To use a PyNode load the script of wanted node in Blender's text editor. After this you need to add Dynamic node to your node setup. The node added will contain a file selector. Use it to select the file loaded in text editor. After you have loaded it, it setups the node. If the code is valid, the node will be ready to use. If not, please check Blender's console to see what is wrong.

Note that if you make any changes to the script you need to press the Update button seen on the Dynamic node.

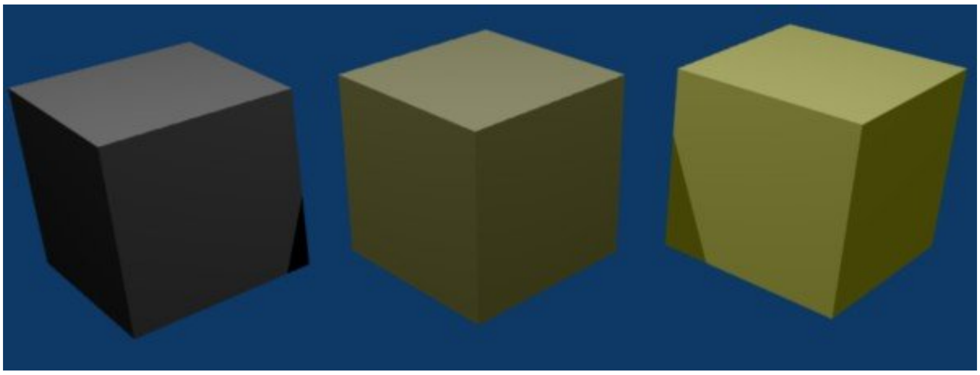
You can find PyNode recipes at the cookbook.

Previous: Manual/Node Materials Contents *Next:* Manual/Ambient Light

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.5x**

Description

Ambient light is all around us and is the result of the sun and lamps scattering photons every which way, reflecting off of and wavelengths being absorbed by objects. Rather than try to calculate the exact intensity of each and every photon, use the Ambient light settings to generally light the scene.



Ambient settings of 0.00, 0.25, and 0.50 receiving a yellow light with RGB values of (0.5,0.5,0.0)

You can specify the amount of Ambient light (the color of which is set in the world settings); use an off−white to simulate the color of the ambient lighting inside closed rooms or on different planets based on its sun's visible spectrum. Generally, about half of RGB gives a nice soft lighting to a scene.

Setting Ambient Light

Suggested Earth Sunny Day Settings

You set the color of ambient light by clicking on the color swatch and using the color picker to choose your color, or by manually adjusting the RGB sliders, or LMB clicking on the slider value (number) and entering a number using your keyboard. These fields are outlined in red in the example above.

This simple example of ambient light settings mimics Earth on a sunny day. The color of the sky at the horizon is white with a touch of haze and a tinge of blue, and the zenith overhead is a dark blue. The sun is out and bright, with a yellow tinge, as on a cloudless day. Think Carribean, and you'll get the idea. London in the fog is gray; night−time is black, etc.

Material Ambient Effect

Each object's material setting (diffuse shader) has an Ambient slider that lets you choose how much ambient light that object receives. Generally, about 0.1 to half is good, depending on the color of the ambient light. A setting of 0.0 means that the object does not receive any ambient light; it is only lit by light that actually gets to it. A setting of 1.0 means that it is lit by all of the world's ambient light. Increasing the Ambient setting has the effect of flattening the shading, since the ambient light washes out the diffuse shading.

You should set the Material slider as shown above in the three cube example, based on the amount of ambient light you think the object will receive. Something deep in the cave will not get any ambient light, whereas something close to the entrance will get more. Note that you can animate this effect, to change it as the object comes out of the shadows and into the light.

World Ambient Occlusion

A trick for games is to illuminate surfaces based on how far they are away from occluding geometry, called ambient occlusion. If looks like ambient light is lighting the objects in the scene, and is the process of Ambient Occlusion. The color of the "light" applied to surfaces is affected by the world Ambient color. See wiki page on Ambient Occlusion.

Radiosity

Radiosity is the process of applying radiated light from objects in the scene as they add to the color of nearby ambient light, which in turn affects the color of the ambient light. See wiki page on Radiosity.

Previous: Manual/Ambient Occlusion	Contents	*Next:* Manual/Exposure

User Manual: Contents | Guidelines | Blender Version **2.44**

Image 1a: Marble Dog with SSS. Watch especially the ears and the paws.

Image 1b: And the same without SSS.

Many organic and some inorganic skins are not totally opaque right at the surface, so light does not just bounce off the top surface. Instead, some light also penetrates the skin surface, and scatters around inside, taking on the color of the insides and emerging back out to blend with the surface reflection. Human/animal skin, the skin of grapes, tomatoes, fruits, wax, gels (like honey, or Jello) and so on all have subsurface scattering (SSS), and photo−realism really cannot be achieved without it.

SSS can be found in the Material buttons (F5), and is limited to diffuse shading only, it does not affect specular shading.

How it works

Image 2: First pass of SSS.

Actually calculating the light path beneath the surface of an object would be practically impossible. But it has been shown that it is not necessary to do this, and that one can use a different approach.

Blender calculates SSS in two steps:

- At first the brightness of the surface is calculated, from the frontside of the object as well as from it's backside. This is pretty much the same as in a normal render. Ambient−Occlusion, Radiosity, the type of diffuse Shader, the light color etc. is taken into account (*Image 2*).
- In the second step the image is rendered finally, but now the SSS shader replaces the diffuse shader. Instead of the lamps the calculated lightmap is used. The brightness of a surface point is the calculated "Average" of the brightness of it's surrounding points. Depending on your settings the whole surface may be taken into account, and it's a bit more complicated than simply calculating the average, but don't bother to much with the math behind it.

Instead let's see what SSS does to a distinct light point.

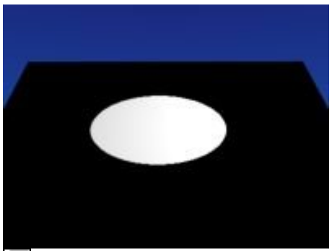


Image 3a: No SSS.

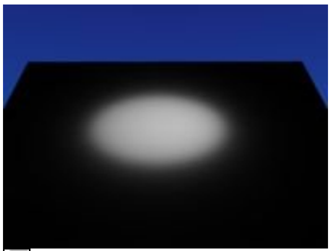


Image 3b: Small SSS radius.

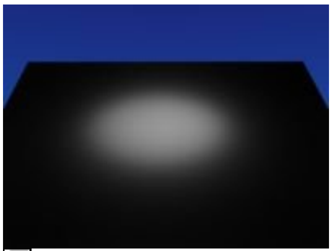


Image 3c: SSS radius enlarged.

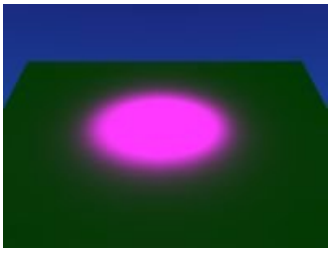


Image 3d: SSS with very large green radius value.

If you turn on SSS the light is distributed over a larger Area. The size of this area depends on the radius values. Instead of distributing all colors with the same amount, you may choose different radius values for each of the RGB−Colors.

If you use a very large radius value for a color, it's light is evenly distributed over the whole object.

Enabling SubSurface Scattering

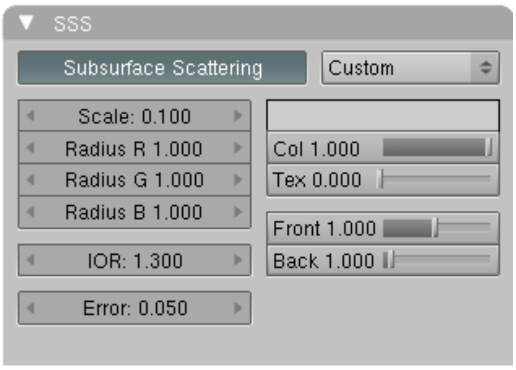


Image 4: The SSS Panel. SSS is already enabled.

- Enable SSS by clicking on the Subsurface Scattering button. You will see your preview panel render change somewhat, as additional processing kicks in.
- Various pre−sets are defined for you, selected by clicking the Selector to the right of Custom. If you don't like any of them, you can define a Custom set. When you select a pre−set, the *Radius* values, the color and the *IOR* are set for you. The remaining options are not set (because they are mostly dependent on the size of your object).

SubSurface Scattering doesn't need raytracing. But since it is dependent on the incident light and shadows, you need proper shadow calculation (which may need raytracing).

Options

The numeric sliders control how the light is scattered:

Scale
: The scale of your object, in Blender units, across which you want the scattering effect to take place. For the presets scale **1.0** means **1** Blender unit equals **1** millimeter, scale **0.001** means **1** Blender unit equals **1** meter.

Radius R, Radius G and Radius B
: The light blurring radius. As the light travels through the object and back up to emerge from the surface at some other point, it creates a path length. These sliders allow you to adjust the average length of that path. The longer the path length is, the more evenly this color is distributed.

IOR
: The IOR value determines the falloff of incident light. Higher values means that light falls off faster. The effect is quite subtle and changes the distribution function only a little bit. By the examination of many different materials a value of **1.3** to **1.5** have been found fitting.

Error
: This parameter controls how precisely the algorithm samples the surrounding points. Leaving it at **0.05** should give images without artifacts. It can be set higher to speed up rendering, potentially with errors. Setting it at **1.0** is a good way to quickly get a preview of the look, with errors.

The color swatch and blend control the color of the SSS shader.

<swatch>
: This has two effects:
    1. If you think of the SSS as a strange sort of lamp, this would be the lights color.
    2. It also affects the scattering − the darker the color the more the light is scattered.

  So if you set it to green, the lit areas of the object will appear in green, and green is scattered only a little. Therefore the darker areas will appear in red and blue. You can compensate the different scattering by setting a larger radius for the color.

Col
: This controls how much the R, G, B option modulates the diffuse color and textures. Note that even with this option set to **0.0**, the R, G, B option still influences the scattering behavior.

Tex
: How much the surface texture is blurred along with the shading.

Front

Factor to increase or decrease the frontscattering. When light enters through the front of the object, how much is absorbed or added? (Normally **1.0** or **100%**).

Back

Factor to increase or decrease the backscattering. Light hitting an object from behind can go all the way through the object and come out on the front of the object. This happens mostly on thin objects, like hands and ears.

Developing your own SSS material

Follow these simple steps to make your own SSS material:

- Set the SSS color on a value of your choice, normally the predominant color of the object. If you want to use different radiuses for the colors, don't make it to dark.
- Set the scale factor. If you want to see much translucency you need small objects or large scale values.
- Set the radius values.
- Adjust the brightness with the Front and Back values.

Example: Grapes

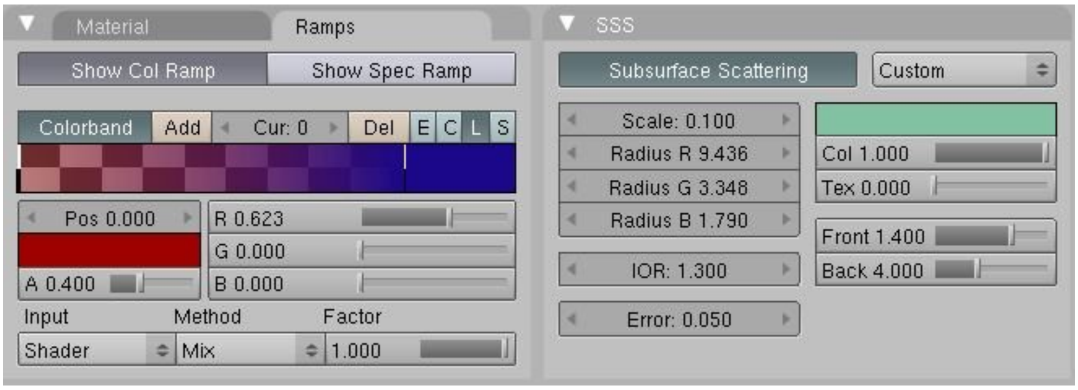


Image 5: Subsurface Settings for the Grapes in Image 6.

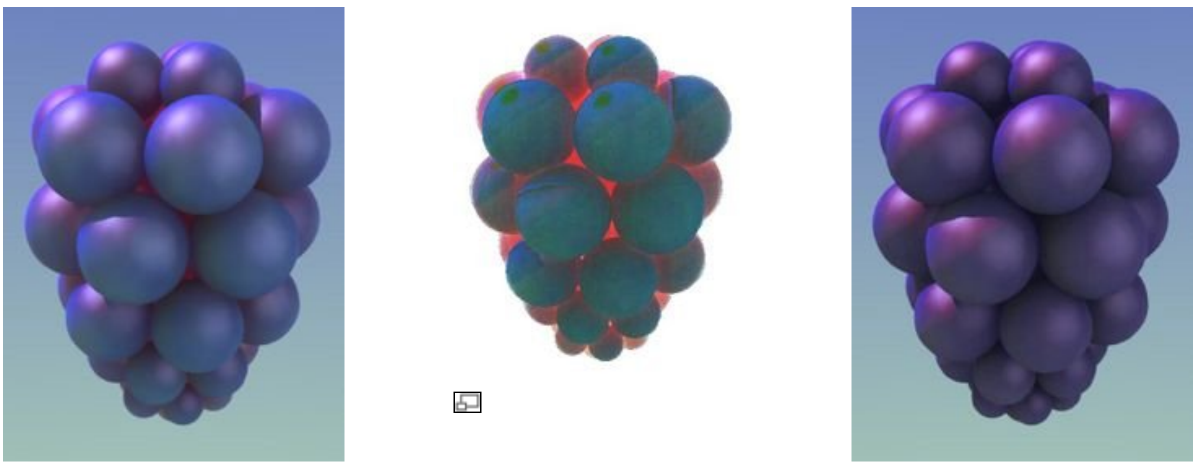


Image 6a: With SSS.

Image 6b: Difference between 6a and 6c, with strongly enhanced brightness and contrast.

Image 6c: Without SSS.

The skin of the grape is a purple colorramp, and we observe that grapes have a fairly red specular glow. The scene is lit with a bright sun from above and behind, and a wide soft area light as a key light. A cloud texture is used to introduce surface variations.

In the example in (*Image 6a*), we have SSS turned on to give a green color based on the inside of a grape. The red *Radius*−Value is quite large, the green *Radius*−Value is larger than the blue one. We can observe the effects of these settings in (*Image 6b*). Though the SSS color is green, the green values are only increased at the very bright spots on the grapes. Green and blue are nearly equally scattered (the larger radius for green compensates the green SSS−color). Since red is scattered very much, red is missing on the parts that are lit from front. The red light is scattered all over each grape, so the same amount of light is emitted from a larger area, partially from the backside of the grapes.

Where we see the backsides of the grapes (pointing away from the light) they appear in red. This has two reasons:

1. Red is scattered stronger than green and blue, so more of the red light reaches the backside of the grapes.
2. The *Back*−Light setting is strongly increased. The *Front*−Light setting is slightly raised to compensate the loss in brightness from the scattering.

Get the .blend file

Example: Skin

Skin is the holy grail of materials, because it is so varied, so imperfect, and so complex. A good skin render is a combination of procedural, UV−mapped images for color, normal, specularity, ambient, and so on. This example uses SSS to get you started.

The model was a human **1.75 BU** high (each BU=1m in real world). We wanted a Caucasian human, so we started with a light tan base material, very little hardness and specularity. For SSS, we started with the "Skin 1" preset. The head was **0.25 BU** in diameter, hence the SSS Scale of **0.150**, because we do not want light from one side to light up the other; there is supposed to be a skull in there!

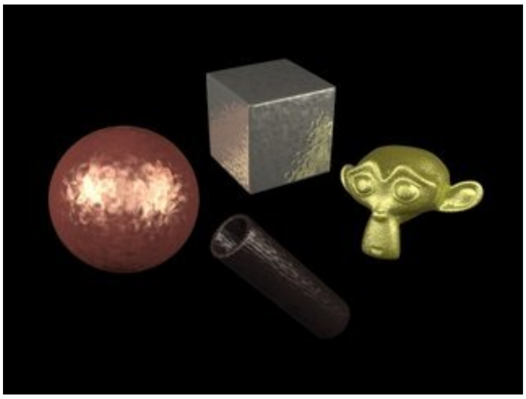
Lighting plays an important part in getting the basic skin to look right. For this example I used a 3−point studio rig:

- Key: Spot placed **5 BU** from subject. Energy **2.0**, Falloff **5.0**, color (**0.98**, **0.98**, **1.0**).
- Fill lights: Hemi placed **5 BU** out to the side, **1 BU** in front of subject. Energy **0.5**, Falloff **10**, color white.

Previous: Manual/Ambient Light	Contents	*Next:* Manual/Textures

User Manual: Contents | Guidelines | Blender Version **2.45**

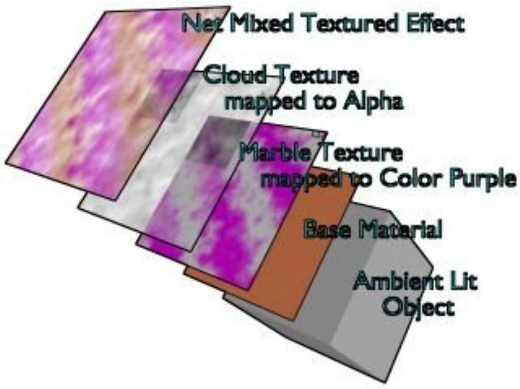
Introduction



Some Metal Textures

The material settings that we've seen so far produce smooth, *uniform* objects, but such objects aren't particularly true to reality, where uniformity tends to be uncommon and out of place. In order to deal with this unrealistic uniformity, Blender allows the user to apply *textures* which can modify the reflectivity, specularity, roughness and other surface qualities of a material.

Textures fall into three primary categories: images, procedural textures (generated by a mathematical formula), and environment maps (used to create the impression of reflections and refractions). For an overview you may want to read our tutorial Using Textures.



Textures Layer on base Material

Textures are like additional layers on top of the base material. Textures affect one or more aspects of the object's net coloring. The net color you see is a sort of layering of effects, shown in this example image. The layers, if you will, are:

1. Your object is lit with **ambient** light based on your world settings.
2. Your base **material** colors the whole surface in a uniform color that reacts to light, giving different shades of the diffuse, specular, and mirror colors based on the way light passes through and into the surface of the object.

3. We have a primary **texture** layer that overlays a purple marble coloring.
4. We next have a **second cloud texture** that makes the surface transparent in a misty/foggy sort of way by affecting the Alpha value
5. These two textures are **mixed** with the base material to provide the net effect; a cube of purplish−brown fog.

Previous: Manual/Material Nodes	Contents	*Next:* Manual/Texture Channels

User Manual: Contents | Guidelines | Blender Version **2.41**

Texture Channels

Mode: All Modes

Panel: Shading/Material Context Texture

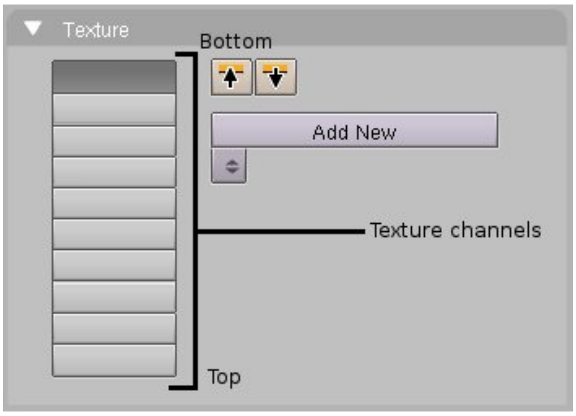
Hotkey: F5

Description

A material can contain 1 to 10 Texture Channels, which are layered in a list. The relationship between a material and a texture is called 'mapping,' and this relationship is two−sided. Each texture channel can contain a texture, and each has individual options for how it is positioned on the object's geometry, and how that material affects the final shading.

Each texture channel has its own individual texture mapping. By default, textures are executed one after another −−Bottom to Top−− layered upon each other. For example, the second channel overlays on top of the first channel which could potentially replace the first channel.

Options



Empty Texture Panel

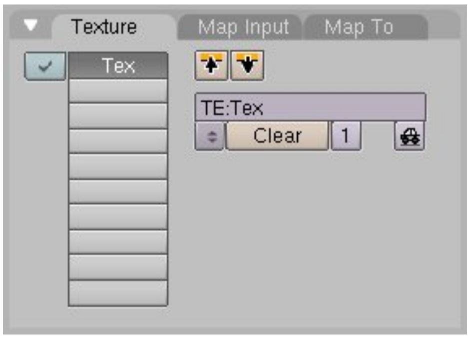
Empty Channels

A channel can be activated by clicking on it in the list. If there is no texture currently in the active texture channel, the following options are available:

Add New
: Add a new texture to the active texture channel

Select an existing texture
: 
: Choose an existing texture from a list, to insert in the active texture channel.

Channels with Textures



Texture Panel with one texture

If the active texture channel contains a texture, further options are made available in the Texture panel, and in two additional panels, Map Input and Map To.

(Texture Panel with one texture) shows the texture with the two new panels. These panels are organized in the sequence in which the texture pipeline is performed. For example, information about the mapping of the texture on the geometry goes *in* through the Map Input panel and then goes "out" the pipeline, as colours of the material, from the Map To panel.

TE:
: The texture itself is designated by its name here, which you can edit here.

Select an existing texture
: Same as for #Empty Channels.

Clear
: Clears the texture from the active channel. The texture data still remains and can be added back to the same channel or another channel from the existing texture selection menu

Number of users (number field)
: The number of materials that use the texture. You can't create a "Single User" copy here, you have to do that in the Texture buttons (F6).

Auto (car icon)
: Automatically generates a (hopefully relevant) name for the texture

Copying and Pasting Texture Channels

By adding an existing texture to a channel, you create a link to that texture, but all the mapping options remain as they are. To copy all texture settings, including the contents of the Map Input and Map To panels, you can copy a given texture channel and paste it into another by using the copy and paste buttons.

Copying and Pasting is used when you want to combine two similar textures where each is slightly different from the other. Rather than manually duplicating the properties of one into another just copy and paste.

Copy (Arrow up)
: Copies the active texture channel's settings into a temporary buffer.

Paste (Arrow down)

Pastes the copied texture channel information into the active texture channel.

Previous: Manual/Textures | Contents | *Next:* Manual/Map Input

User Manual: Contents | Guidelines | Blender Version **2.45**

Texture Map Input

Mode: All Modes

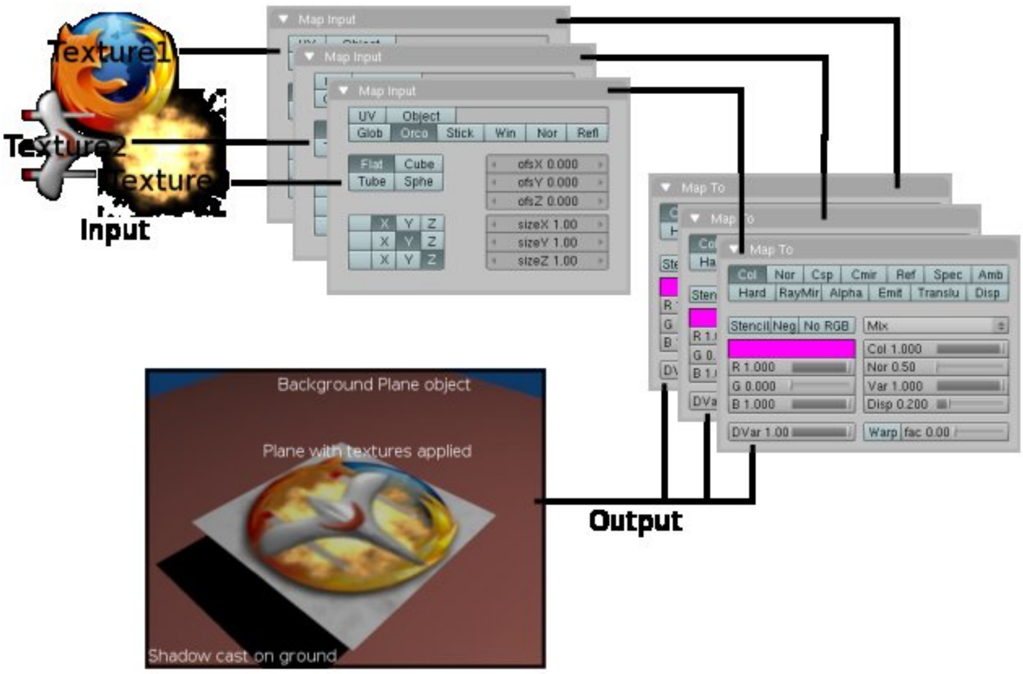
Panel: Shading/Material Context Map Input

Hotkey: F5

Description

Textures need mapping coordinates, to determine how they are applied to geometry. The mapping specifies how the texture will ultimately *wrap* itself to the object. For example, a 2D image texture could be configured to wrap itself around a cylindrical shaped object.

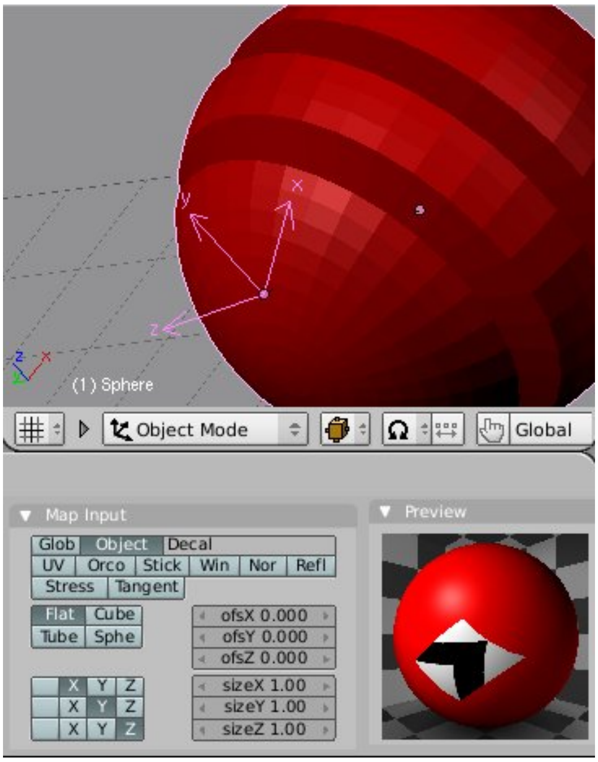
Examples



Conceptual texture pipeline

(*Conceptual texture pipeline*) is what is happening if you could "see" each panel(s) associated with each texture or image at the same time. However, only one panel is visible at any one time. To change a texture's Map Input properties you must first select the texture on the Texture panel.

Position a Decal on a Mesh



Another common use of Map Input, briefly mentioned above, is placing a decal somewhere on a mesh:

- Insert an Empty into your scene, positioned close to the surface of your mesh where you want the center of the decal to be, and oriented with its X−Y axis as you want your decal to appear (the Z axis should point away from the mesh, i.e. normal to the mesh).
- Assign an image texture to the mesh, loading the decal image, and set UseAlpha and ClipCube in the Map Image panel.
- In the Map Input field, select Object and in the blank field to the right of the button, enter the name of the Empty, which by default is "Empty"

In the example to the right, an Empty named "Decal" is just above the surface of the red ball, with the Z pointing away from the ball. In the material settings for the ball, an image of an arrow is mapped to the location and XY orientation of the empty.

You may Scale the empty to make the decal larger or smaller, and/or use the Size sliders in the Map Input panel. You can tweak the position of the decal by moving the empty or using the Offset sliders.

Hints

When choosing a mapping style consider the following questions:

- What is the source of the input Coordinates?
- Where is the top left corner?
- Where does the Texture begin?
- How large is the Texture and how often does it have to be repeated?
- Do we have multiple points to begin the texture, like at a cube?
- Will the Texture be rotated?

Mapping an image to the Window coordinates will cause the image to be tiled across the surface, without stretching, facing the camera without regard to the object's geometry.

Global coordinates, when mapped to Cube, tile the virtual space with the image. Wherever the object happens to be, it gets the texture that exists in that space.

Textures scale along with the object. If you scale an object larger, the image or procedural texture is scaled up as well. To make more patterns of the texture repeat across the surface, either increase the image repeat value (if an image) or increase the SizeXYZ values in the Map Input panel. This can be a pain if you have a texture such as a brick, and you make the wall longer; the bricks will stretch out. You can either use a procedural plug−in texture that figures out the repeats for you, or get out the calculator and divide the "true" texture size by the actual wall size to determine how many repeats there should be.

Use multiple texture channels of the same texture, each Sized differently, to create the natural patterns found in nature (principle of the fractals).

To make them completely independent from the object's scale, either use Global mapping or Object, referencing an empty at the object's center with a scale of 1 (preferably parented to the object if you are animating it.).

Input Source

Mode: All Modes

Panel: Shading/Material Context Map Input

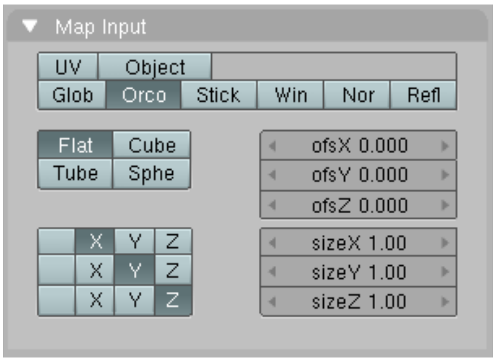
Hotkey: F5

Description

Mapping works by using a set of coordinates to guide the mapping process. These coordinates came come from anywhere, usually the object to which the texture is being applied to.

For UV or Object mapping, enter the name of the UV Texture or Object that you want to use as the orientation for the texture in the little blank field immediately to the right of the UV and Object buttons. The default name for a UV Texture is "UV Tex"; the exact name is found in the Materials panels where the UV Textures are listed. You can only specify one UV Texture for each mapped Texture image. If you want to layer UV Textures, you have to use multiple channels. See UV Unwrapping for more information on using UV Textures.

Options



Map Input Panel

UV
: UV mapping is a very precise way of mapping a 2D texture to a 3D surface. Each vertex of a mesh has its own UV co−ordinates which can be unwrapped and laid flat like a skin. You can almost think of UV coordinates as a mapping that works on a 2D plane with its own local coordinate system to the plane on which it is operating on. This mapping is especially useful when using 2D images as textures, as seen in [UV Mapping](). You can use multiple textures with one set of UV Co−ordinates.

Some Modifiers Prevent UV Mapping

In particular, the Decimate Modifier, even if it is only in the Editing modifier list (stack) and not actually applied to the mesh, prevent UV mapping, since it affects the number of vertices and thus UV coordinates.

To use a UV map to guide the texture placement, click *UV* and enter the name of the UV Texture in the *input field*. The name must match exactly and is case−sensitive. If the name does not match one of the existing UV Textures, the field will be red.

Then, for the Color of the texture, map the image texture to Col (if the image is the color/diffuse image). Map to Nor if the image is the bump map, etc. See Bump and Normal mapping for more info on those special kinds of images. See the next page for more info on Map To panel.

Object
: Uses a child Object's texture space as source of coordinates. The Object name must be specified in the text button on the right. Often used with Empty objects, this is an easy way to place a small image as a logo or decal at a given point on the object. This object can also be animated, to move a texture around or through a surface

Glob − Global
: The scene's Global 3D coordinates. This is also usefull for animations; if you move the object, the texture moves across it. It can be useful for letting objects appear or disappear at a certain position in space.

Orco − Original Coordinates
: "Original Co−ordinates" − The object's local texture space. This is the default option for mapping textures.

Stick
: Uses a mesh's sticky coordinates, which are a form of per−vertex UV co−ordinates. If you have made *Sticky* co−ordinates first (F9 Mesh Panel, *Sticky* Button), the texture can be rendered in camera view (so called "Camera Mapping").

Win − Window
: The rendered image window coordinates. This is well suited to blending two objects.

Nor − Normal
: Uses the direction of the surface's normal vector as coordinates. This is very useful when creating certain special effects that depend on viewing angle

Refl − Reflection

Uses the direction of the reflection vector as coordinates. This is useful for adding reflection maps − you will need this input when Environment Mapping.

2D to 3D Mapping

Mode: All Modes

Panel: Shading/Material Context Map Input

Hotkey: F5

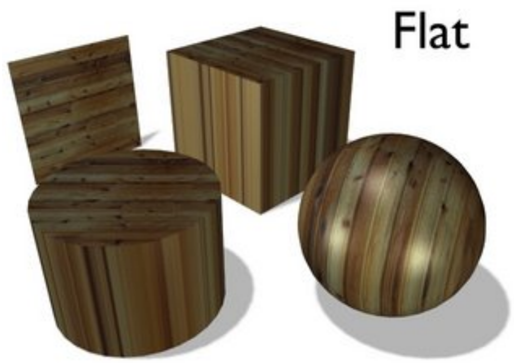
Description

The Image texture is the only true 2D texture, and is the most frequently used and most advanced of Blender's textures. Because images are two−dimensional, the way in which the 2D texture coordinate is translated to 3D must be specified in the *mapping* buttons.

Options

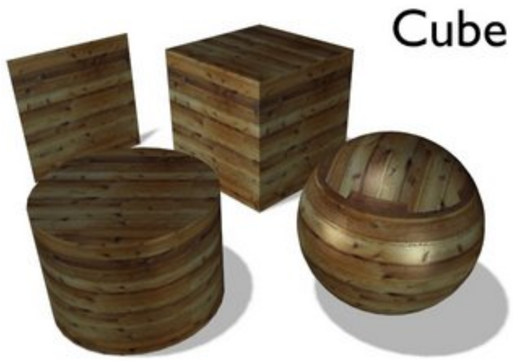
Depending on the overall shape of the object, one of these types may be more useful than others.

If the UV button (see #Input Source; *Others also?*) is enabled you can imagine all these buttons except for the Flat button to be disabled.



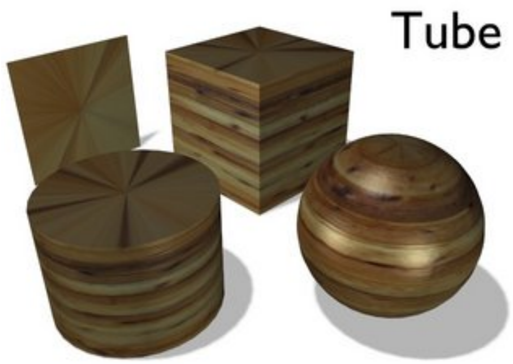
Flat Mapping.

Flat
: Flat mapping gives the best results on single planar faces. It does produce interesting effects on the sphere, but compared to a sphere−mapped sphere the result looks flat. On faces that are not in the mapping plane the last pixel of the texture is extended, which produces stripes on the cube and cylinder.



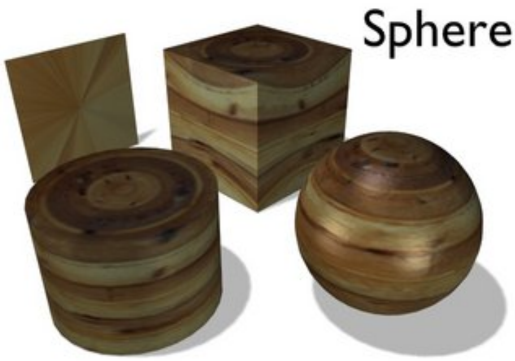
Cube Mapping.

Cube
: Cube mapping often gives the most useful results when the objects are not too curvy and organic (notice the seams on the sphere).



Tube Mapping.

Tube
: Tube mapping maps the texture around an object like a label on a bottle. The texture is therefore more stretched on the cylinder. This mapping is of course very good for making the label on a bottle or assigning stickers to rounded objects. However, this is not a cylindrical mapping so the ends of the cylinder are undefined.



Sphere Mapping.

Sphere
: Sphere mapping is the best type for mapping a sphere, and it is perfect for making planets and similar objects. It is often very useful for creating organic objects. It also produces interesting effects on a cylinder.

Co−ordinate Offset, Scaling and Transformation

Mode: All Modes

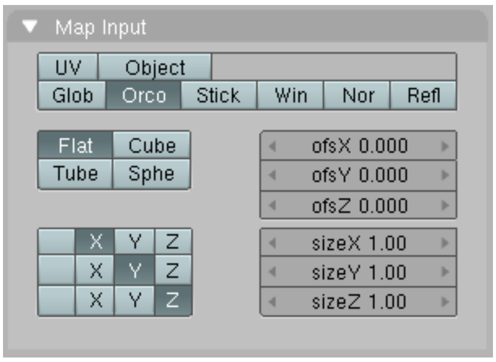
Panel: Shading/Material Context Map Input

Hotkey: F5

Description

For extra control, the texture space can also be tweaked, to move, scale and flip the apparent texture on its axes

Options



Map Input Panel

ofsX, ofsY, ofsZ − Offset
: The texture co−ordinates can be translated by an offset. Enlarging of the Ofs moves the texture towards the top left.

sizeX, sizeY, sizeZ − Size
: The scale of the texture space. Enlarging the texture space will make the apparent texture scale smaller. The texture is as often repeated (if set up as repeating) as sized here.

[ ], X, Y, Z (x3) − Axes
: Re−orders the X, Y and Z coordinates. You can flip axes to mirror a texture, or ignore axes all together.

3D View texture space transform

Mode: Object Mode

Hotkey: T

Description

The texture space can also be transformed interactively in the 3D View, just like moving or scaling an object. This determines the area that the texture uses to define its co−ordinates.

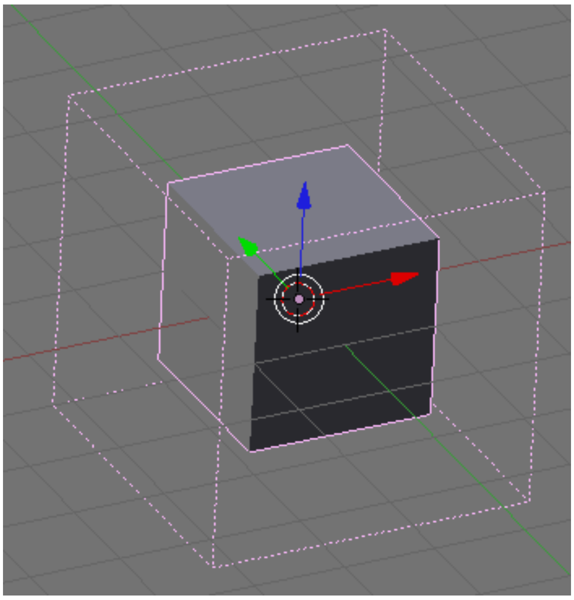
Options

Texture Space popup menu

Press T while an object is selected to get the Texture Space popup menu. The following options are available:

Grab/Move
: Moves the objects' texture space

Scale
: Scales the object's texture space



Scaled Texture Space box.

Reset Texture Space

This manual editing overrides the automatic calculation of texture space based on the object's local co−ordinates and size. To return to automatic texture space calculation, enable the AutoTexSpace button in the Link and Materials Panel in the Editing context (F9)

Previous: Manual/Texture Channels	Contents	*Next:* Manual/Map To

User Manual: Contents | Guidelines | Blender Version **2.45**

Texture Map To

Mode: All Modes

Panel: Shading/Material Context Map To

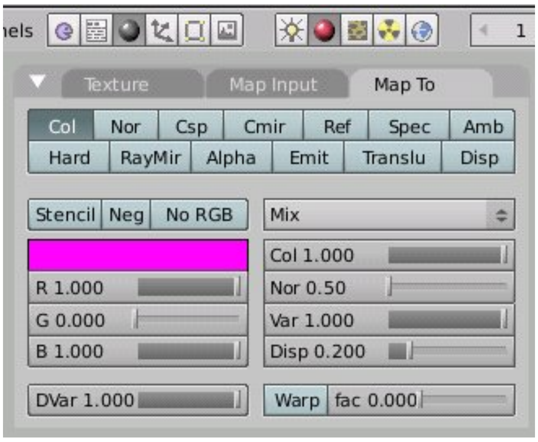
Hotkey: F5

Description

Not only can textures affect the color of a material, they can also affect many of the other properties of a material. The different aspects of a material that a texture influences are controlled in the Map To panel.

The result is also dependent from the type of input value. A texture may carry Intensity Information (0−255), Transparency (0.0−1.0), RGB Color (3 channels of 0.0−1.0) or Normal Vectors (either Bump or real Normals, see Section Bump and Normal Maps).

Options



Map To Panel.

Aspect

Some of the following options use three−state toggle buttons, meaning that the texture can be applied as positive or negative to some aspect of the material, or not at all. All of these buttons are independent.

Col (on/off)
: Influences the Material's RGB color

Nor (+/−/off)
: Commonly called bump mapping, this alters the direction of the surface normal. This is used to fake surface imperfections or unevenness via bump mapping, or to create reliefs.

Csp (on/off)
: Influences the Specular color, the color of the "reflects" created by the lamps on a glossy material.

Cmir (on/off)

Influences the mirror color. This works with environment maps and raytraced reflection

Ref (+/−/off)
: Influences the amount of diffuse reflection.

Spec (+/−/off)
: Influences the amount of specular reflection.

Amb (+/−/off)
: Influences the amount of Ambient light the material receives

Hard (+/−/off)
: Influences the specular hardness amount. A DVar of 1 is equivalent to a Hardness of 130, a DVar of 0.5 is equivalent to a Hardness of 65.

RayMir (+/−/off)
: Influences the strength of raytraced mirror reflection

Alpha (+/−/off)
: Influences the Opacity of the material. See Use Alpha for Object Transparency. Also use ZTransp for light and if combining multiple channels.

Emit (+/−/off)
: Influences the amount of light Emitted by the material

Translu (+/−/off)
: Influences the Translucency amount

Disp (+/−/off)
: Influences the Displacement of vertices, for using Displacement Maps.

Channel Filter

Stencil
: The active texture is used as a mask for all following textures. This is usefull for semitransparent textures and "Dirt Maps". See the example below (Stencil). Black sets the pixel to "untexturable".

Neg
: The effect of the Texture is negated. Normally white means on, black means off, Neg reverses that.

No RGB
: With this option, an RGB texture (affects color) is used as an Intensity texture (affects a value)

Color Swatch
: If the texture is mapped to Col, what color is blended in according to the intensity of the texture? Click on the swatch or set the RGB sliders

DVar
: Destination Value (not for RGB). The value with which the Intensity texture blends with the current value. Two examples:

- The Emit value is normally 0. With a texture mapped to Emit you will get maximal effect, because DVar is 1 by default. If you set DVar to 0 no texture will have any effect.
- If you want transparent material, and use a texture mapped to Alpha, nothing happens with the default settings, because the Alpha value in the Material panel is 1. So you have to set DVar to 0 to get transparent material (and of course ZTransp also). This is a common problem for beginners. Or do it the other way round − set Alpha to 0 and leave Dvar on 1. Of course the texture is used inverted then.

Blending Mode

How this channel interacts with other channels below it. See the Compositing Mix node for information and examples on the effect of each mixing mode.

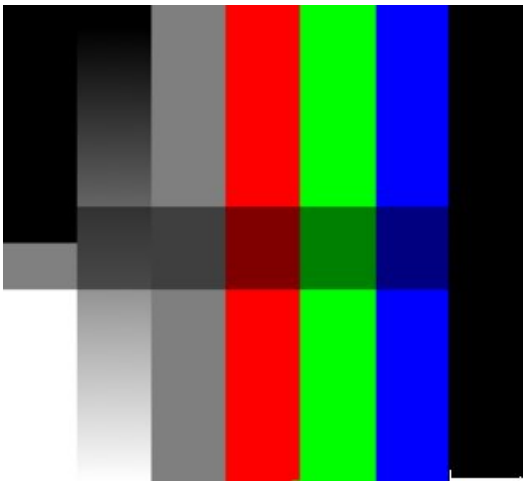
Mix
Add, Subtract, Multiply, Divide

Screen, Overlay, Difference
Darken/Lighten
Hue, Saturation, Value

Impact Sliders

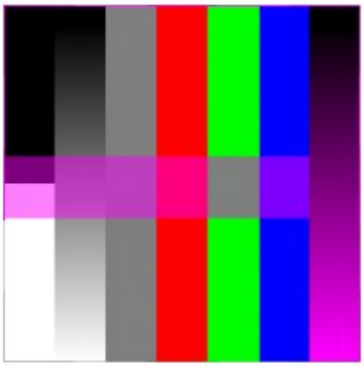
Col
The extent to which the texture affects colour
Nor
The extent to which the texture affects the normal. Affects Normal, Bump and Displacement Maps.
Var
The extent to which the texture affects the other values
Disp
The extent an intensity texture changes the displacement. See section Displacement Maps.
Warp and Fac (factor)
Distort subsequent textures to give an illusion of shape. See (Warp).

Examples



The input file. Vertical: Black, White, Black/White blend, 50% Grey, RGB, Black/Alpha blend. The brighter horizontal bar has a transparency of 50%.

The *input file* demonstrates the impact of different input values. The color of the underlying plane is magenta (R=1.0, G=0.0, B=1.0). The Map To Col color of the texture is yellow (R=1.0, G=1.0, B=0.0).



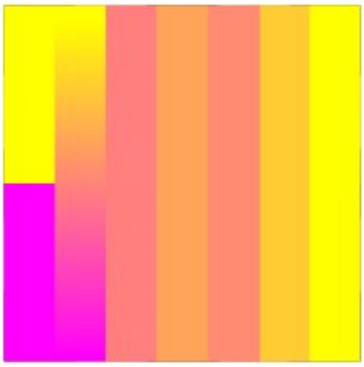
Use Alpha

Normally the color of the texture is opaque. If you Use Alpha with the image texture the alpha information of the image is evaluated. This will not make the material transparent, only the texture! So these pixels show the underlying material color (*Use Alpha*).

No RGB shows the result of the NO RGB option. The RGB texture is used as an Intensity texture. White produces the "Map To" color.

Neg invertes the respective values.

No RGB



No RGB and Neg

Hints

Every texture is opaque to the textures above it. This is no problem, if one texture is for example mapped to color, the other mapped to alpha etc. If you want to give a Mesh multiple materials see instead [Multiple Materials].

Stencil

Mode: All Modes

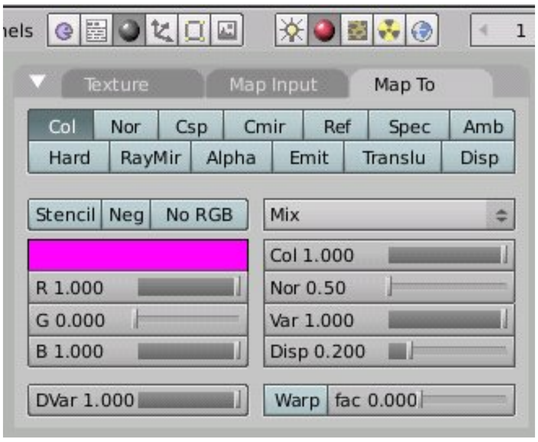
Panel: Shading/Material Context Map To

Hotkey: F5

Description

The Stencil mode works similar to a layer mask in a 2D program. The effect of a stencil texture can not be overridden, only extended. You need an intensity map as input.

Options



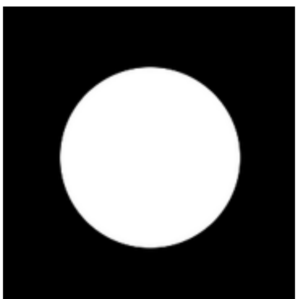
Map To Panel.

Stencil
: The active texture is used as a mask for all following textures. Black sets the pixel to "untexturable".

Examples

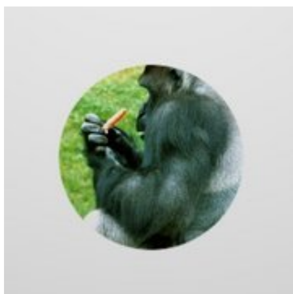
The Stencil mode works similar to a layer mask in a 2D program. The effect of a stencil texture can not be overridden, only extended. You need an intensity map as input.

Where the mask is black the following textures have no effect. Stencil needs intensity as input, so you have to use No RGB if you would like to use either images without alpha or a texture with a colorband (e.g. Blend Textures with a colorband).



The mask

The texture



The result

Stencil map with a radial blend.

You can blend two textures if you use a smooth blend texture as stencil map (*Stencil map with a radial blend*).

Warp

Mode: All Modes

Panel: Shading/Material Context Map To

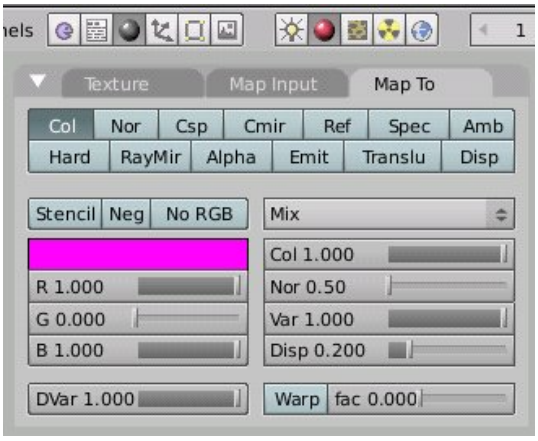
Hotkey: F5

Description

The option Warp allows textures to influence/distort the texture coordinates of a next texture channel.

The distortion remains active over all subsequent channels, until a new Warp has been set. Setting the fac at zero cancels out the effect.

Options



Map To Panel.

Warp
: Enable and disable the warp distortion

Fac

The amount of distortion

Examples

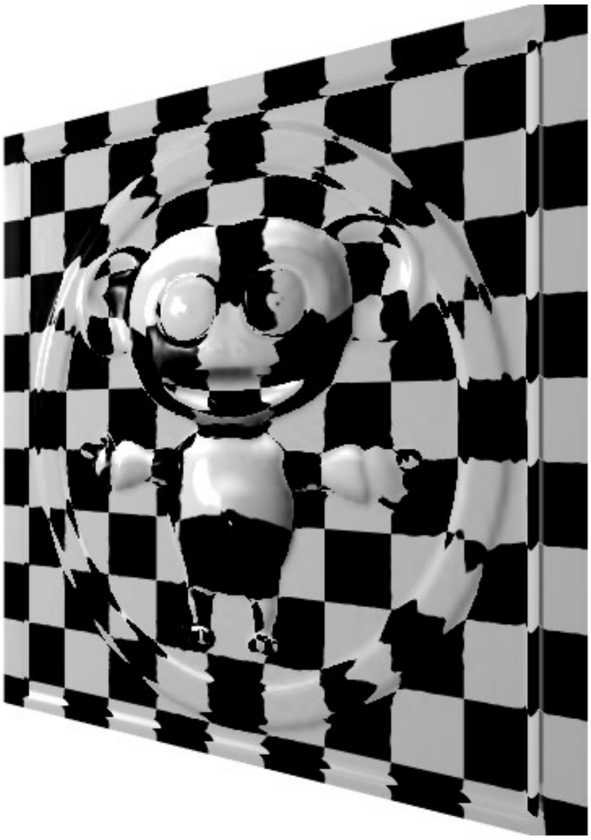
The following example warps a gorilla texture based on a simple blend texture:



The Blend Texture The Texture to warp The warped Result

The same Texture for Normal and Warp mapping.

In this example, the normal map of Cornelius (*The same Texture for Normal and Warp mapping*) was used as normal map as well as warp texture in channel 1. The checkerboard texture is used in channel 2.



Cornelius as a Warp Factor

Previous: Manual/Map Input	Contents	*Next:* Manual/Bump and Normal Maps

User Manual: Contents | Guidelines | Blender Version **2.45**

Bump and Normal Maps

Mode: All Modes

Panel: Shading/Texture Context Image

Hotkey: F6

Description

Normal Maps and Bump Maps both serve the same purpose: they simulate the impression of a detailed 3D surface, by modifying the shading as if the surface had lots of small angles, rather than being completely flat. Because it's just modifying the shading of each pixel, this will not cast any shadows and will not obstruct other objects. If the camera angle is too flat to the surface, you will notice that the surface is not really shaped.

Both bump maps and normal maps work by modifying the normal angle (the direction pointing perpendicular from a face), which influences how a pixel is shaded. Although the terms normal map and bump map are often used synonymously, there are certain differences:

- **Bump maps** are textures that store an **intensity**, the relative height of pixels from the viewpoint of the camera. The pixels seem to be moved by the required distance in the direction of the face normals. You may either use greyscale pictures or the intensity values of a RGB−Texture (including images).
- **Normal maps** are images that store a **direction**, the direction of normals directly in the RGB values of an image. They are much more accurate, as rather than only simulating the pixel being away from the face along a line, they can simulate that pixel being moved at any direction, in an arbitrary way. The drawbacks to normal maps are that unlike bump maps, which can easily be painted by hand, normal maps usually have to be generated in some way, often from higher resolution geometry than the geometry you're applying the map to.

Normal maps in Blender store a normal as follows:

- Red maps from (0−255) to X (−1.0 − 1.0)
- Green maps from (0−255) to Y (−1.0 − 1.0)
- Blue maps from (0−255) to Z (0.0 − 1.0)

Since normals all point towards a viewer, negative Z−values are not stored (they would be invisible anyway). In Blender we store a full blue range, although some other implementations also map blue colors (128−255) to (0.0 − 1.0). The latter convention is used in "Doom 3" for example.

Workflow

The steps involved in making and using Bump and Normal Maps is:

1. Model a highly detailed ("hi−poly") model
2. Bake the Bump and/or Normal maps
3. Make a low−poly, less detailed model
4. Map the map to the low−poly model using a common coordinate system

Consult the Modeling section for how to model a highly detailed model using the Mesh tools. How much detail you put in is totally up to you. The more ridges and details (knobs, creases, protrusions) you put in, the more detailed your map will be.

Baking a map, simply put, is to take the detail of a high polygon mesh, and apply it to a similar object. The similar object is identical to the high−poly mesh except with less vertices. Use the Render Bake feature in Blender to accomplish this.

Modeling a low−poly using Blender's Mesh editing tools. In general, the same or similar faces should exist that reflect the model. For example, a highly detailed ear may have 1000 faces in the high−poly model. In the low−poly model, this may be replaced with a single plane, oriented in the same direction as the detailed ear mesh.

Mapping is the process of applying a texture to the low−poly mesh. Consult the Textures section for more information on applying a texture to a mesh's material. Special considerations for Bump and Normal Maps is:

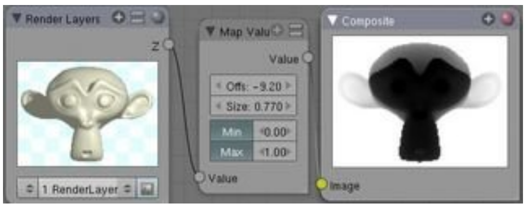
- When using a Bump map, map the texture to *Nor* and enable *No RGB.*
- When using a Normal map, map the texture to *Nor*

The coordinate systems of the two objects must match. For example, if you bake using a UV map of the high−poly model, you must UV map the low poly model and line up its UV coordinates to match the outline of the high−poly image (see UV unwrapping to line up with the high−poly map edges.

Examples

To set up the scene, position an orthographic camera about 10 units from a monkey. UV unwrap the Monkey; in this example we did an easy *Project from View (Bounds)*. Scale the camera so that the monkey fills the frame. Set your format resolution to 512x512 (square textures render faster in game engines, usually). No lights or anything else is required.

Use Render Bake to create a Normal map.



Composite noodle to make Bump Maps

Then, enable *Do Composite* and set up the compositing noodle shown to the right. This is your Bump Map. You make a bump map by using that same orthographic camera in the composite noodle shown. The front of the model is 9.2 units from the camera, and the visible parts are 1.3 units deep (1/1.3=0.77). If you save your image as a JPG, you will have 24−bit depth. If you save it as a half OpenEXR image, 16−bit depth; as a full OpenEXR image, 32−bit depth. Alternatively, you can use the ZUtilz Plugin, however, it only gives you an 8−bit depth and the resolution of the Bump Map may be too small for the relatively great range of Z−Values.

Now we are going to look at some examples. First the render of "Suzanne" (*Suzanne Render*). The second picture shows Suzanne's Normal Map (made with Blender's Normal Baking system. The rightmost picture in the top row shows the *Bump Map of Suzanne*.

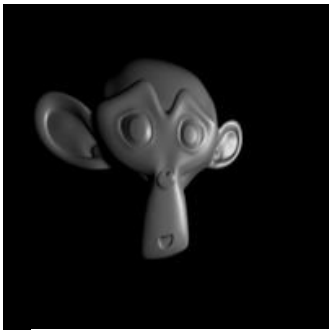
Both maps here are used as textures on a plane, first the Normal Map (*Render of the Normal Map*), then the Bump Map (*Render of the Bump Map*). In both cases the camera stayed in the same position in which the Maps were made (perpendicular to the plane).

Suzanne Render

Normal Map of Suzanne

Bump Map of Suzanne

Side view render of the Normal Map applied to a plane



Side view render of a Normal Map made with an Ortho camera, applied to a plane. The same camera position as in the previous picture, less perspective distortion.

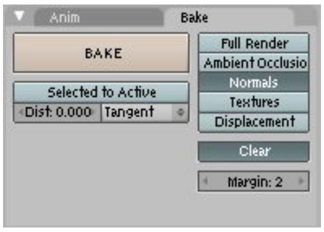
Render of the Normal Map applied to a plane, perpendicular to the surface.

Render of the Bump Map applied to a plane.

The *Render of the Normal Map* is only pseudo 3D. You can't look at the side of the head (*Side view render*).

If you use an Ortho camera to create the Normal Map, you will get less perspective distortion (*Render with an Ortho camera*).

Baking Options



The Render Bake panel

Render baking is hyperlinked here. In summary, there Blender supports normal map baking, amongst other bakeable things:

- Tangent: Bakes a normal map that is independent of view, and is thus the best choice for animation.
- Object: Bakes normals in the object's coordinates, and can be moved, but not deformed.
- Camera: Bakes normals with the old method, meaning you cannot deform or move the object.
- World: Bakes normals with coordinates, but object can't be moved or deformed.

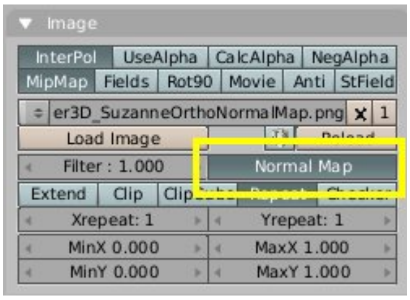
Other options include:

- Selected to active: This option will bake the selected mesh's normals

to the active mesh (e.g. the high poly mesh is the selected and is baked to the low−poly mesh which is the active).

- Distance: A parameter that controls the max distance from one object to another.
- Clear: Wipes the current image clear of any data.
- Margin: The amount (in pixels) of which to extend the normal map.

Using And Creating Normal Maps



Normal Map Button in the Image Panel

Normal and Bump Maps are simple to use. Make sure you apply the Texture in the Material buttons Map To panel to Nor. The strength of the effect is controlled with the NumButton Nor on the same Panel.

If you want to use a Normal Map, you have to set the Normal Map button in the Image Panel in the Texture buttons F6 (*Normal Map Button*).

Since only the normals get affected during render, you will not get shadow or AO or other '3D' effects. It's still just a texture.

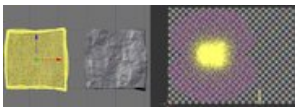
To bake normal maps is a different process, but simple.

1) Make a mesh.

2) Create a low polygon version of the object either by lowering the multires level (if you are using multires), by using a decimate modifier, or by just modeling the same mesh with less vertices. For simplicity, we'll say the low polygon object= mesh L and the high polygon object= mesh H.

3)To make this all work, mesh L needs to have a UV image. Assign seams and unwrap to a new image of the desired size (with UV test grid enabled. This will be explained in the "Common Problems" section further down the page).

You'll notice a new "32 bit float" option under the new image panel. This



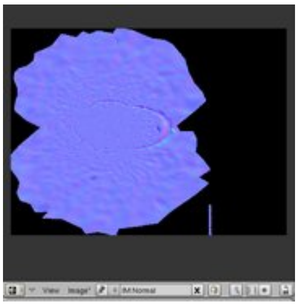
The low−poly mesh unwrapped

option creates your image in 32 bits, which is useful when generating displacement maps. If saving the 32−bit image, it must be saved to a format supporting 32 bits, such as OpenEXR.

4)Before you bake the normal maps, both meshes must be in the exact same location, so mesh L must be moved back to mesh H's location with ALT + G.

5)If the "Selected to active" button is enabled (which it should), you will need to select first mesh H, and then mesh L.

6) Go to the "Scene" tab (F10) and then to the bake tab (next to "anim"). Select "Normals", and customize your options according to your needs. Then press the big shiny "Bake" button and watch your the map appear in all it's glory.



The normal map after baking

7) Nope, you're not done yet. We still need to add the normal map as a texture. The next step is to go into the materials tab (F5) and apply a new texture. Select the type as "Image",

and select your normal map texture (hopefully you named it) with the arrows.

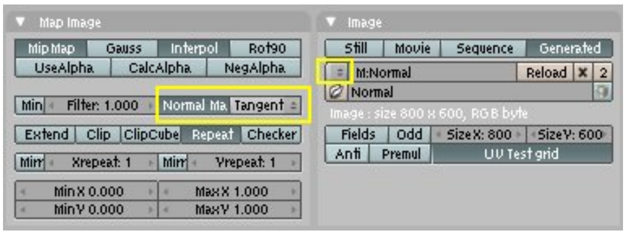


Image selected

8) Back in the main material panel under "Map Input", change the texture coordinates to UV. Under "Map To", check "Nor" and deselect "Col". Set the "Nor" value to 1 (Or a different value, depending on how strong you want it). You're done!

Previous: Manual/Map To	Contents	*Next:* Manual/Displacement Maps

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.37a**

Displacement Maps

Panel: Material buttons Map To

Hotkey: F5

Description

Displacement mapping allows a texture input to manipulate the position of vertices on rendered geometry. Unlike Normal or Bump mapping, where the shading is distorted to give an illusion of a bump (discussed on the previous page), Displacement Maps create real bumps, creases, ridges, etc in the actual mesh. Thus, the mesh deformations can cast shadows, occlude other objects, and do everything that changes in real geometry can do.

Options

Settings for a Displacement Map.

The strength of the displacement is controlled with the number fields Disp and Nor.

- If a texture provides only normal information (e.g. Stucci), vertices move according to the texture's normal data. The normal displacement is controlled by the Nor slider.
- If a texture provides only intensity information (e.g. Magic, derived from color), vertices move along the directions of their normals (a vertex has no normal itself, it's the resulting vector of the adjacent faces). White pixels move outward in the direction of the normal, black pixels move in the opposite direction. The amount of displacement is controlled with the Disp slider.

The two modes are not exclusive. Many texture types provide both information (Cloud, Wood, Marble, Image). The amount of each type can be mixed using the respective sliders. Intensity displacement gives a smoother, more continuous surface, since the vertices are displaced only outward. Normal displacement gives a more aggregated surface, since the vertices are displaced in multiple directions.

The depth of the displacement is scaled with an object's scale, but not with the relative size of the data. This means if you double the size of an object in object mode, the depth of the displacement is also doubled, so the relative displacement appears the same. If you scale inside editmode, the displacement depth is not changed, and thus the relative depth appears smaller.

Hints

Displacement maps move the render faces, not the physical mesh faces. So, in 3D View the surface may appear smooth, but render bumpy. To give a detailed surface, there has to be faces to displace and have to be very small. This creates the tradeoff between using memory and CPU time versus render quality.

From best to worst, displacement works with these object types using the methods listed to control the render face size:

- **SubSurfaced Meshes**

  Render face size is controlled with render subsurf level. Displacement really likes smooth normals.
- **Manually (editmode) subdivided meshes**

  Control render faces with number of subdivides. (This can be combined with the above methods.) Displaces exactly the same Simple Subsurf, but slows editing down because of the OpenGL overhead of drawing the extra faces. (You can't turn the edit subdivide level down this way).
- **Meta Objects**

  Control render faces with render wiresize. Small wire == more faces.

The following are available, but currently don't work well. It is recomended that you convert these to meshes before rendering.

- **Open NURBS Surfaces**

  Control render faces with U/V DefResolu. Higher numbers give more faces. (Note normal errors).
- **Closed NURBS Surfaces**

  Control with DefResolu control. (Note the normal errors, and how implicit seam shows).
- **Curves and Text**

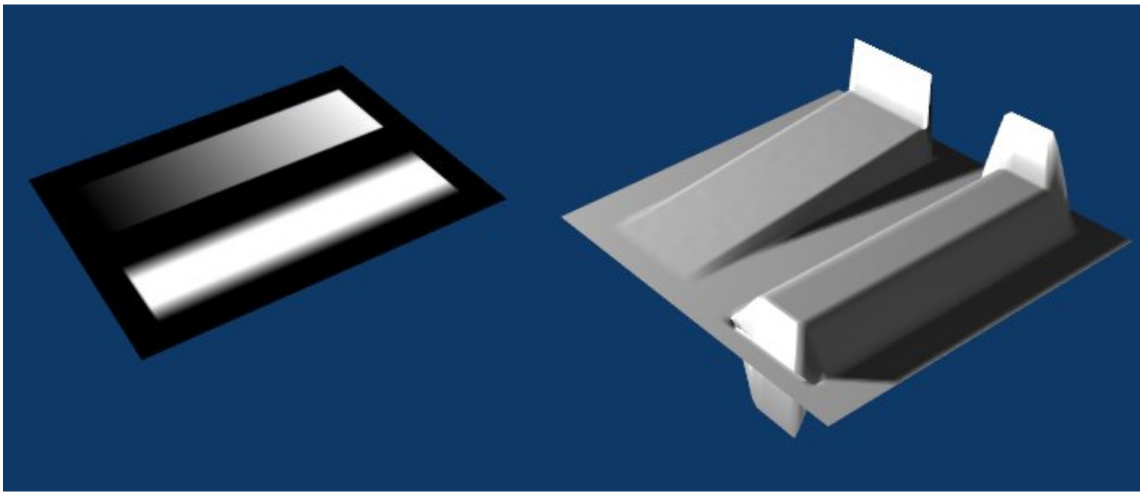
  Control with DefResolu control. Higher gives more render faces. (Note that the large flat surfaces have few renderfaces to displace).

Using the Displace Modifier

If you want more control over your displacement, you'll probably want to use the Displace Modifier. This feature has lots of different options so that you can customize the displacement excactly to your liking.

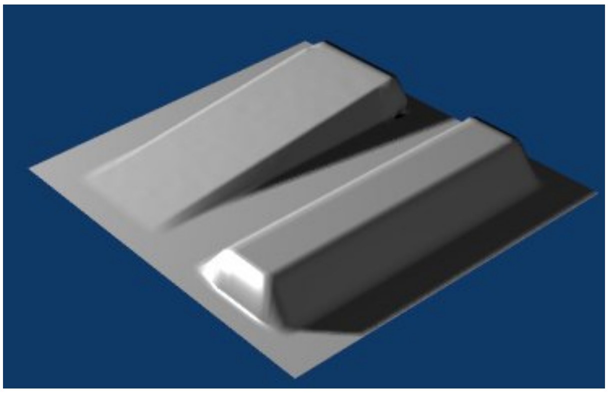
Examples

At first a not−so−well working example (*Texture and Displacement Map*):



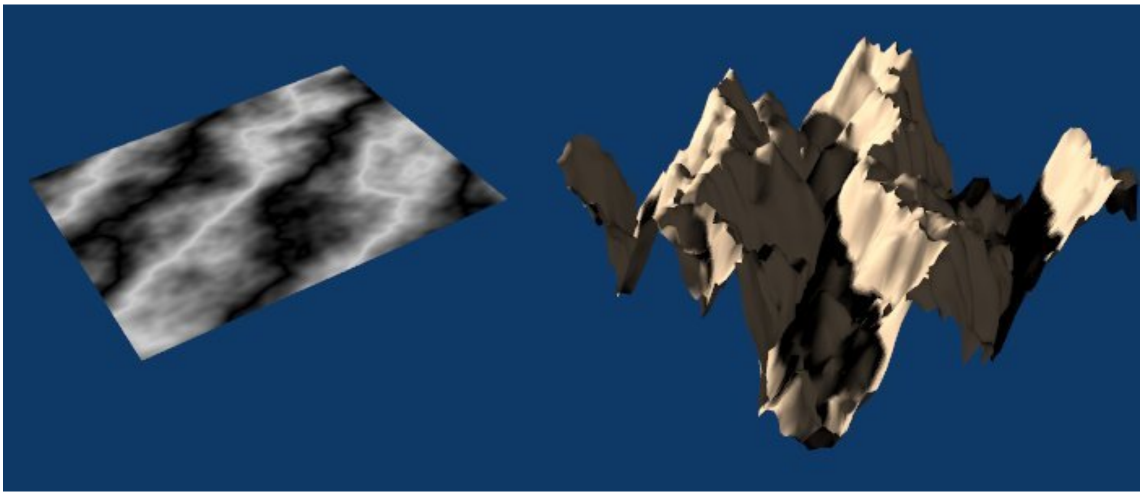
Texture and Displacement Map. The result shows some errors.

The sharp contrasting transitions from black to white yield problems. To correct this, use a little bit of gaussian blur on the texture (*A blurred Texture*).



A blurred Texture yields the correct result.

If you use a Texture (like Marble) with no sharp transitions, the displacement works quite well (*A Displacement Map to create a landscape*).



A Displacement Map to create a landscape.

Advanced Materials often use Displacement Maps. Here a Marble Texture was applied to various Map To values, including Disp. The brink of the "comet" would be flat otherwise (*A Displacement Map for advanced materials*). The sphere has 1024 faces.

A Displacement Map for advanced materials.

See Also

- Bump and Normal Maps

How to create a Displacement Map

If you are using procedural textures, start with a flood of 50% gray as this neutral color will make a good base for your texture, as it does not cause any displacement. Some adjustment can be done using the Colors Panel Bright and Contr sliders if you desire. Continue shaping your map with blacks and whites with the Bright and Contr sliders. Done? Then back to the "Map To" panel (if you are not using a displace modifier) and check "Disp" and tweak the amount.

Sharp lines in Displacement Maps can cause normal problems, since a renderface can be requested to move only one of its verts a great distance relative to the other 2−3. You tend to get better results if a small gaussian blur is run on the image first.

Previous: Manual/Bump and Normal Maps	Contents	*Next:* Manual/Texture Options

User Manual: Contents | Guidelines | Blender Version **2.45**

Texture Buttons

Mode: All Modes

Panel: Shading/Textures Context

Hotkey: F6

Description

Once a new texture has been added to a material, it can be defined by switching to the Texture Buttons (F6) or  sub−context of the Shading context to obtain *Texture buttons*.

A new, empty texture Button Window presents two panels:

1. a Texture Preview and
2. a Texture panel, the latter with two tabs.

Texture Preview

Mode: All Modes

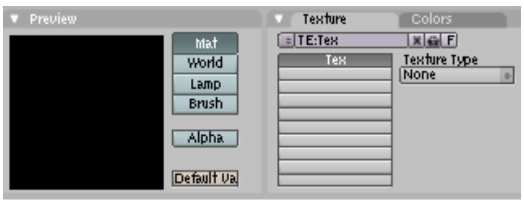
Panel: Shading/Textures Context Preview

Hotkey: F6

Description

The texture preview panel provides a quick previsualisation of how the texture looks on it's own, without mapping.

Options



Texture buttons

You can choose what kind of textures you are editing:

Material
: Edit the stack of textures linked to the active material

World
: Edit the stack of textures linked to the active world

Lamp
: Edit the stack of textures linked to the active lamp

Brush
: Edit the stack of textures linked to the active brush (in sculpt mode).

Alpha
: Show alpha in preview

Default Va
: Reset all the texture properties to their default values

Texture Channels

Mode: All Modes

Panel: Shading/Textures Context Texture

Hotkey: F6

Description

This panel lets you manage the list of texture channels available for the particular shading data that you are working on (material, world or lamp).

Options

Standard datablock selector
: Choose, rename, unlink, automatically generate a name, or add a fake user for the active texture

Texture channel button
: Change the active texture channel

Texture Types.

Here you can also define the active texture's type. The available types of textures are:

Image
: Allows an image to be loaded and used as a texture. See Image Textures

EnvMap
: To simulate Reflections (and Refractions) without Raytracing. See Environment Maps.

Plugin
: Allows for loading an external piece of code to define the texture. See Texture Plugins.

Procedural
: The remaining options define 3D procedural textures, which are textures that are defined mathematically and are built into Blender. See Procedural Textures.

Texture Colors

Mode: All Modes

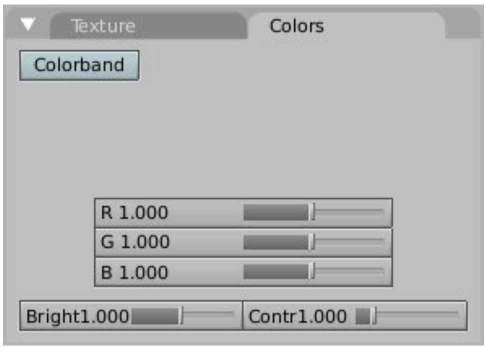
Panel: Shading/Textures Context Colors

Hotkey: F6

Description

All textures may be modified by the Bright(ness) and Contr(ast) buttons in the Colors panel. All textures which posess RGB−Values − including Images and Environment Maps − may be modified with the RGB sliders. (*Texture Colors Panel*).

Options



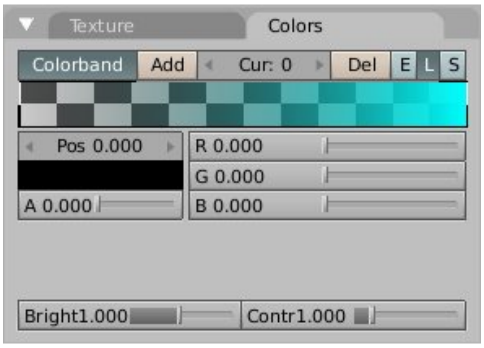
Texture Colors Panel.

R, G, B
: Tint the color of a texture by brightening each red, green and blue channel

Brightness
: Change the overall brightness/intensity of the texture

Contrast
: Change the contrast of the texture

Colorbands



Texture Colorband.

If intensity−only textures are used, the result is a black and white texture, which can be greatly enhanced by the use of colorbands. The colorband is an often−neglected tool in the Colors tab in the Texture Panel that gives you an impressive level of control over how procedural textures are rendered. Instead of simply rendering each texture as a linear progression from 0.0 to 1.0, you can use the colorband to create a gradient which progresses through as many variations of color and transparency (alpha) as you like (*Texture Colorband.*).

To use Colorbands, select a procedural texture, such as Wood. Click the Colorband button. The Colorband is Blender's gradient editor. Each point on the band can be placed at any location and can be assigned any color and transparency. Blender will interpolate the values from one point to the next.

For information on using the colorband UI controls, see Colorbands in the Ramps section of this manual.

Hints

- The alpha slider changes the transparency of the selected arrow of the colorband.
- If you use a colorband, the results of the texture are intensity and RGB. The alpha values deliver intensity, the colors RGB. Use the *NoRGB* button in the Materials context to calculate intensity from the RGB values.

See Also

- Ramps

Previous: Manual/Displacement Maps | Contents | *Next:* Manual/Procedural Textures

User Manual: Contents | Guidelines | Blender Version **2.44**

Procedural Textures

Mode: All Modes

Panel: Shading/Textures Context

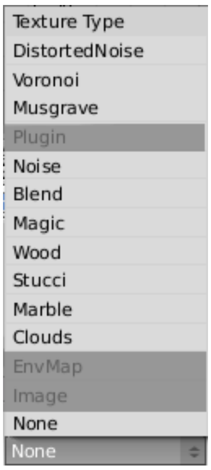
Hotkey: F6

Description

Procedural textures are textures that are defined mathematically. They are generally relatively simple to use, because they don't need to be mapped in a special way − which doesn't mean that procedural textures can't become very complex.

These types of textures are 'real' 3D. By that we mean that they fit together perfectly at the edges and continue to look like what they are meant to look like even when they are cut; as if a block of wood had really been cut in two. Procedural textures are not filtered or anti−aliased. This is hardly ever a problem: the user can easily keep the specified frequencies within acceptable limits.

Options



The Texture Type list in the Texture panel of the Texture Buttons (F6).

The non procedural textures are greyed out in *The Texture Type list*.

Nabla
: Almost all procedural textures in Blender use derivatives for calculating normals for texture mapping (with as exception Blend and Magic). This is important for Normal and Displacment Maps. The strength of the effect is controlled with the Nabla Number Button.

Hints

- Use the size buttons in the Map Input Panel to set the size that Procedural Textures are mapped to.
- Procedural textures can either produce colored textures, intensity only textures, textures with alpha values and normal textures. If intensity only ones are used the result is a black and white texture, which can be greately enhanced by the use of colorbands. If on the other hand you use colorbands and need an intensity value, you have to switch on No RGB in the Map To panel.

Noise Basis

Mode: All Modes

Panel: Shading/Texture Context

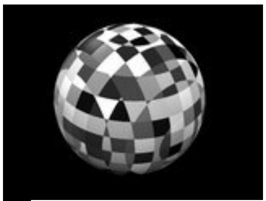
Hotkey: F6

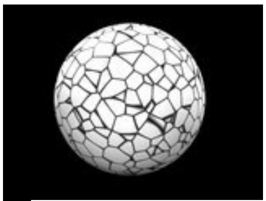
Description

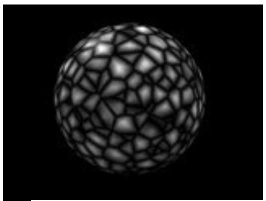
Each noise−based Blender texture (with the exception of Voronoi and simple noise) has a Noise Basis setting that allows the user to select which algorithm is used to generate the texture. This list includes the original Blender noise algorithm. The Noise Basis settings makes the procedural textures extremely flexible (especially Musgrave).

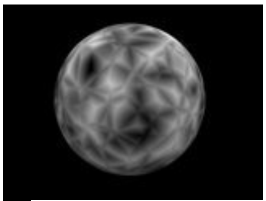
Examples

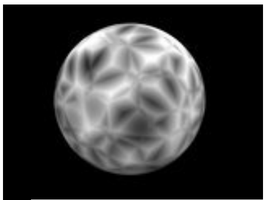
The Noise Basis governs the structural appearance of the texture.

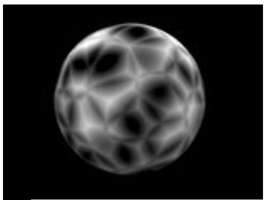

Cellnoise

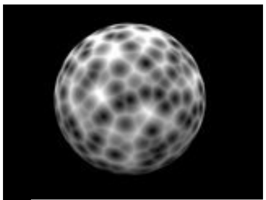

Voronoi Crackle

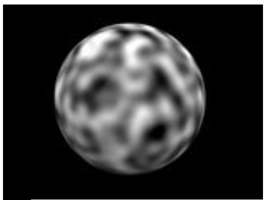

Voronoi F2−F1


Voronoi F4


Voronoi F3


Voronoi F2


Voronoi F1


Blender Original

There are two more possible settings for Noise Basis, which are relatively similar to Blender Original:

- Improved Perlin

- Original Perlin

Clouds

Mode: All Modes

Panel: Shading/Texture Context Clouds

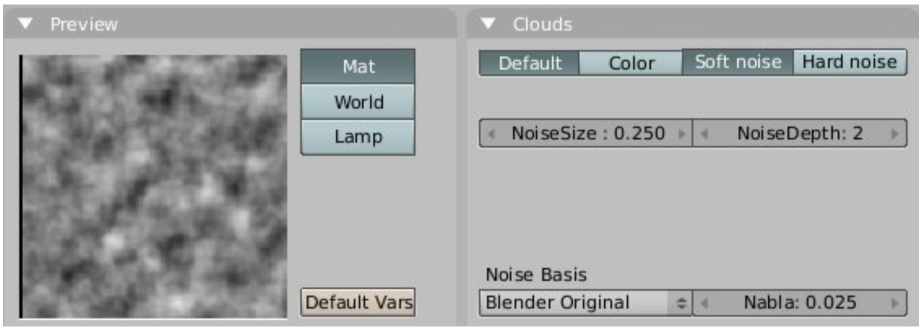
Hotkey: F6

Description

Often used for: Clouds, Fire, Smoke. Well suited to be used as Bumpmap, giving an overall irregularity to the material.

Result(s): Intensity (Default) or RGB−Color (Color)

Options



Clouds Texture Panels.

Default
: The standard Noise, gives an Intensity.

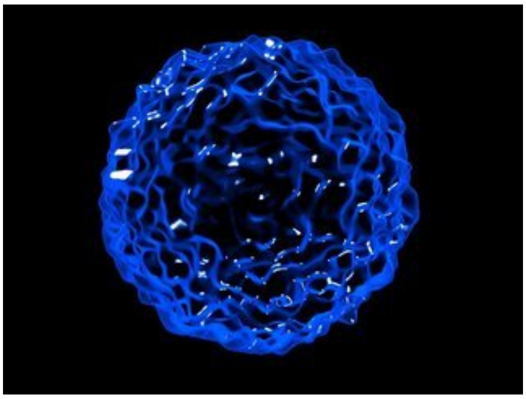
Color
: The Noise gives an RGB value.

Soft Noise/Hard Noise
: Changes the contrast and sharpness

NoiseSize
: The dimension of the Noise table.

NoiseDepth
: The depth of the Cloud calculation. A higher number results in a long calculation time, but also in finer details.

Examples



A Clouds Texture was used to displace the surface.

Technical Details

A three−dimensional table with pseudo random values is used, from which a fluent interpolation value can be calculated with each 3D coordinate (thanks to Ken Perlin for his masterful article "An Image Synthesizer", from the SIGGRAPH proceedings 1985). This calculation method is also called Perlin Noise. In addition, each noise−based Blender texture (with the exception of Voronoi and simple noise) has a new "Noise Basis" setting that allows the user to select which algorithm is used to generate the texture.

Marble

Mode: All Modes

Panel: Shading/Texture Context Marble

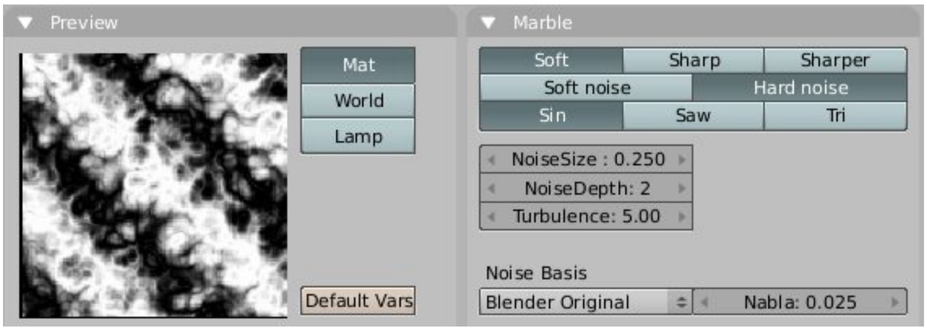
Hotkey: F6

Description

Often used for: Marble, Fire, Noise with a structure.

Result(s): Intensity value only.

Options



Marble Texture Panels

Soft/Sharp/Sharper
: Three pre−sets for soft to more clearly defined Marble.

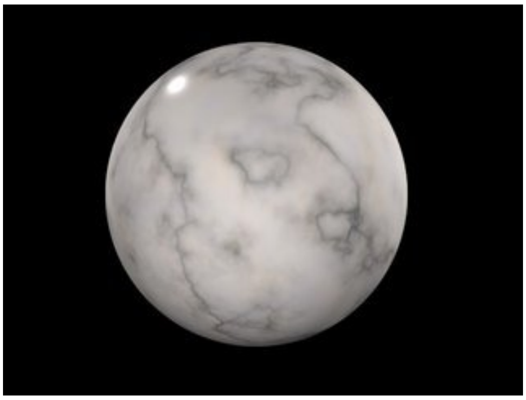
Soft Noise/Hard Noise
: The Noise function works with two methods.

NoiseSize
: The dimensions of the Noise table.

NoiseDepth
: The depth of the Marble calculation. A higher value results in greater calculation time, but also in finer details.

Turbulence
: The turbulence of the sine bands.

Examples



Marble from the Blender Materials Library

Technical Details

Bands are generated based on a sine formula and Noise turbulence.

Stucci

Mode: All Modes

Panel: Shading/Texture Context Stucci

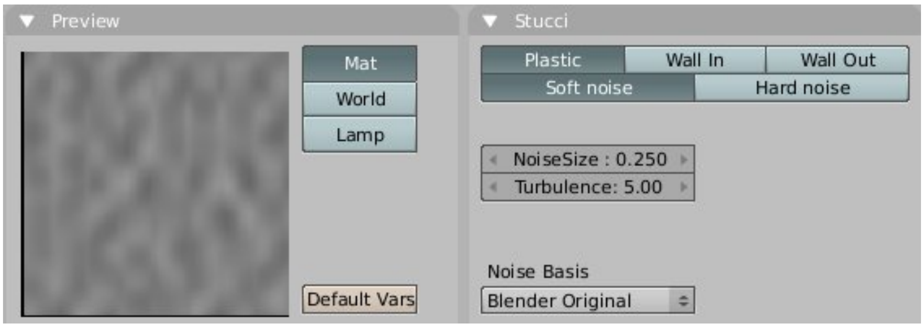
Hotkey: F6

Description

Often used for: Stone, Asphalt, Oranges. Normally for Bump−Mapping to create grainy surfaces.

Result(s): Normals and Intensity

Options



Stucci Texture Panels

Plastic
: The standard Stucci.

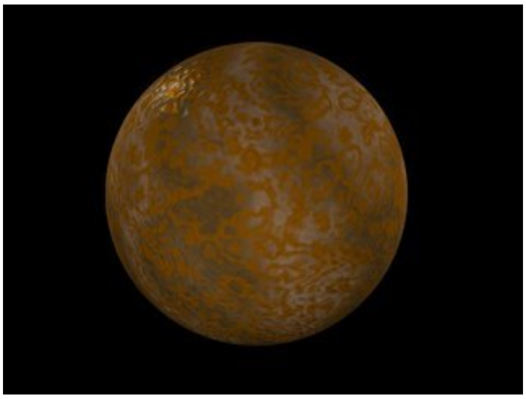
Wall In, Wall out
: This is where Stucci gets it name. This is a typical wall structure with holes or bumps.

Soft Noise/Hard Noise
: There are two methods available for working with Noise.

NoiseSize
: The dimension of the Noise table.

Turbulence
: The depth of the Stucci calculations.

Examples



Some rusty metal. Stucci was used to "Bump" the surface a bit.

Technical Details

Based on noise functions

Wood

Mode: All Modes

Panel: Shading/Texture Context Wood

Hotkey: F6

Description

Often used for: Wood

Result(s): Intensity only

Options

Wood Texture Panels.

Bands
    The standard Wood texture.
Literal|Rings
    This suggests 'wood' rings.
BandNoise
    Gives the standard Wood texture a certain degree of turbulence.
RingNoise
    Gives the rings a certain degree of turbulence.
Soft Noise/Hard Noise
    There are two methods available for the Noise function.
NoiseSize
    The dimension of the Noise table.
Turbulence
    The turbulence of the BandNoise and RingNoise types.

Examples

See the section Tutorials/Textures/Wood for a method to create procedural wood.

"Wenge Wood" by Claas Eike Kuhnen.

Colorbands are used in both materials and textures, as well as other places where a range of colors can be computed and dipslayed. In the example to the right, we want to texture a snake, specifically the deadly coral snake. We want to make a repeating set of four colors: Black, yellow, red, yellow (and then back to black again). We also want to make the rings sharp in definition and transition. This example uses 8 color band settings: 0 and 7 are black; 1 and 2 are yellow, 3 and 4 are red, and 5 & 6 are yellow. Position 0 and 1 close together, 2 and 3, etc. Use a little noise and turbulence; together with the scales texture you should get really close!

Technical Details

Generation
: In this case, bands are generated based on a sine formula. You can also add a degree of turbulence with the Noise formula.

Magic

Mode: All Modes

Panel: Shading/Texture Context Magic

Hotkey: F6

Description

Often used for: This is difficult, it was hard to find an application whatsoever. One could use it for "Thin Film Interference", if you set Map Input to Refl and use a relatively high Turbulence.

Result(s): RGB

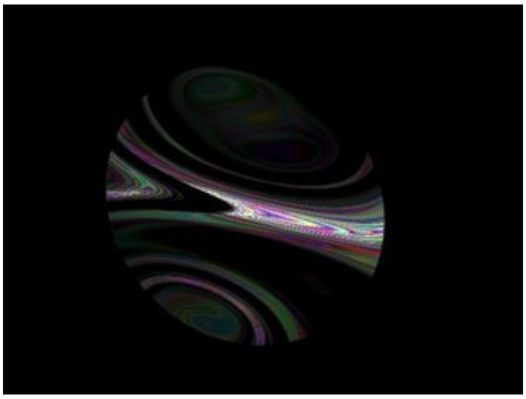
Options

Magic Texture Panels.

Depth
: The depth of the calculation. A higher number results in a long calculation time, but also in finer details.

Turbulence
: The strength of the pattern.

Examples



"Thin Film Interference" with Magic Texture

I've used two Magic Textures in *"Thin Film Interference" with Magic Texture*. Both use the same texture with Depth 4, Turbulance 12. Both have Map Input set to Refl. The first texture is mapped to Nor, the second to Col.

Technical Details

The RGB components are generated independently with a sine formula.

Blend

Mode: All Modes

Panel: Shading/Texture Context Blend

Hotkey: F6

Description

Often used for: This is one of the most important procedural textures. You can use blend textures to blend other textures together (with Stencil), or to create nice effects (especially with the Map Input: Nor trick). Just remember: if you use a colorband to create a custom blending, you may have to use No RGB, if the Map To value needs an intensity input!

Result(s): Intensity

Options

Blend Texture Panels

Lin
: A linear progression.

Quad
: A quadratic progression.

Ease
: A flowing, non−linear progression.

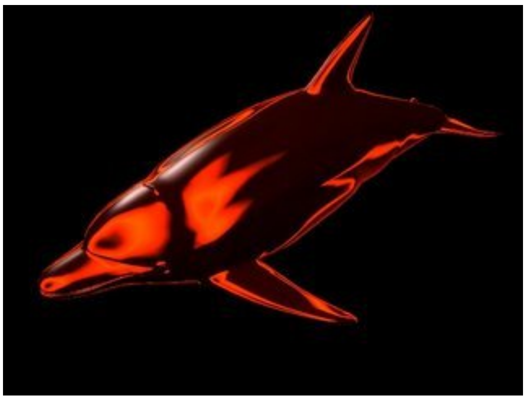
Diag
: A diagonal progression.

Sphere
: A progression with the shape of a three−dimensional ball.

Halo
: A quadratic progression with the shape of a three−dimensional ball.

Flip XY
: The direction of the progression is flipped a quarter turn.

Examples



A custom radial blend with Map Input set to Nor, Map To Ref and Emit.

Technical Details

The Blend texture generates a smoothly interpolated progression.

Noise

Mode: All Modes

Panel: Shading/Texture Context Noise

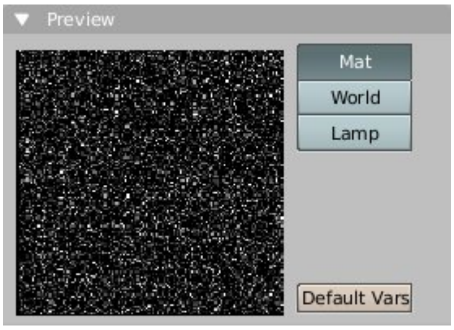
Hotkey: F6

Description

Often used for: White noise in an animation. This is not well suited if you don't want an animation. For material roughness take clouds instead.

Result(s): Intensity

Options



Noise Texture Panel.

There is no panel and no buttons. Just switch it on.

Technical Details

Although this looks great, it is not Perlin Noise! This is a true, randomly generated Noise. This gives a different result every time, for every frame, for every pixel.

Musgrave

Mode: All Modes

Panel: Shading/Texture Context Musgrave

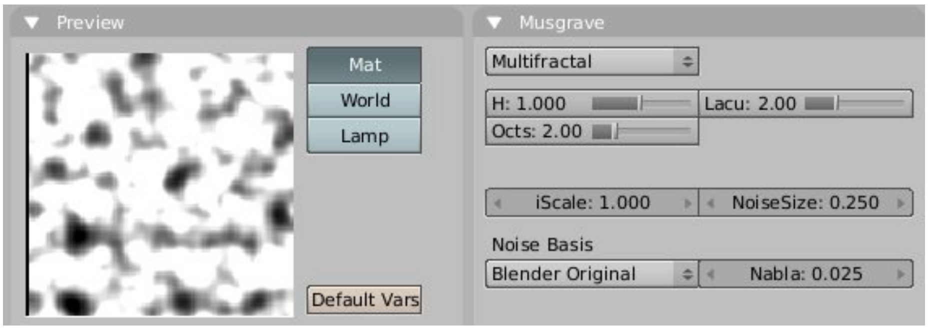
Hotkey: F6

Description

Often used for: Organic materials, but it's very flexible. You can do nearly everything with it.

Result(s): Intensity

Options



Musgrave Texture Panels.

Noise Types
: This procedural texture has five noise types on which the resulting pattern can be based and they are selectable from a dropdown menu at the top of the tab. The five types are:
    - ◊ fBm:
    - ◊ Hetero Terrain:
    - ◊ Hybrid Multifractal:
    - ◊ Ridged Multifractal:
    - ◊ Multifractal:

  These noise types determine the manner in which Blender layers successive copies of the same pattern on top of each other at varying contrasts and scales.

In addition to the five noise types, Musgrave has a noise basis setting which determines the algorithm that generates the noise itself. These are the same noise basis options found in the other procedural textures.

The main noise types have four characteristics which can be set in the number buttons below the dropdown list. They are:

H (Fractal Dimension) − Range 0 to 2)
: Fractal dimension controls the contrast of a layer relative to the previous layer in the texture. The higher the fractal dimension, the higher the contrast between each layer, and thus the more detail shows in the texture.

Lacu (Lacuniarity) − Range 0 to 6)
: Lacuniarity controls the scaling of each layer of the Musgrave texture, meaning that each additional layer will have a scale that is the inverse of the value which shows on the button. i.e. Lacu = 2 −> Scale = 1/2 original

Octs (Octave) − Range 0 to 8)
: Octave controls the number of times the original noise pattern is overlayed on itself and scaled/contrasted with the fractal dimension and lacuniarity settings.

The Hybrid Multifractal, Ridged Multifractal, and Hetero Terrain types have additional settings:

Ofst (Fractal Offset)

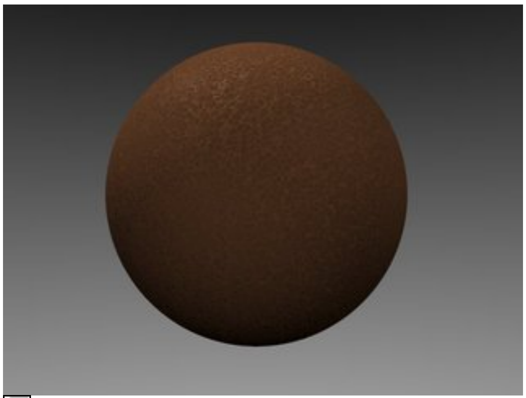
All three have a "Fractal Offset" button labeled Ofst. This serves as a "sea level" adjustment and indicates the base height of the resulting bump map. Bump values below this threshold will be returned as zero.

Gain

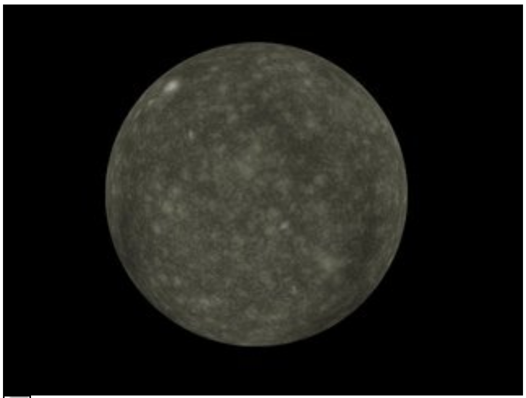
Hybrid Multifractal and Ridged Multifractal both have a Gain setting which determines the range of values created by the function. The higher the number, the greater the range. This is a fast way to bring out additional details in a texture where extremes are normally clipped off.

Examples

See the Samples Gallery from the release notes for more examples.



Leather made with a single Musgrave Texture



Stone made with a combination of 3 different Musgrave Textures.

Technical Details

More information about these textures can be found at the following URL: Musgrave Documentation

Voronoi

Mode: All Modes

Panel: Shading/Texture Context Voronoi

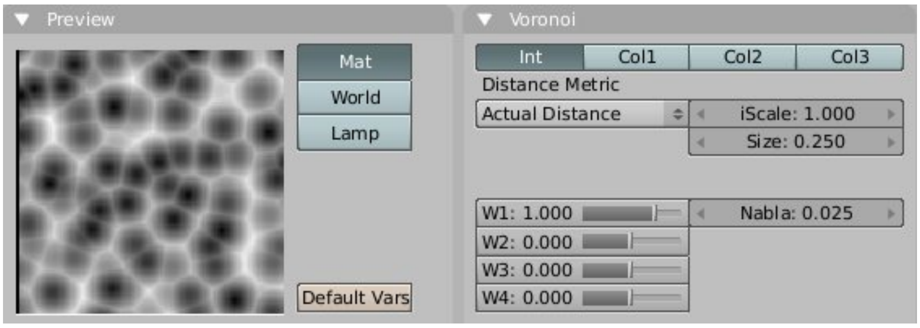
Hotkey: F6

Description

Often used for: Very convincing Metal, especially the "Hammered" effect. Organic shaders (e.g. scales, veins in skin).

Result(s): Intensity (default), Color.

Options



Voronoi Texture Panels.

Distance Metric
: This procedural texture has seven Distance Metric options. These determine the algorithm to find the distance between cells of the texture. These options are:
  - ◊ Minkovsky
  - ◊ Minkovsky 4
  - ◊ Minkovsky 1/2
  - ◊ Chebychev
  - ◊ Manhattan
  - ◊ Distance Squared
  - ◊ Actual Distance

  The Minkovsky setting has a user definable value (the Exp button) which determines the exponent (`e`) of the distance function $(x^e + y^e + z^e)^{1/e}$. A value of one produces the Manhattan distance metric, a value less than one produces stars (at **0.5**, it gives a Minkovsky 1/2), and higher values produce square cells (at **4.0**, it gives a Minkovsky 4, at **10.0**, a Chebychev). So nearly all Distance Settings are basically the same − variations of Minkowsky.

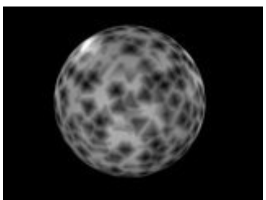
You can get irregularly−shaped rounded cells with the Actual Distance/Distance Squared options.

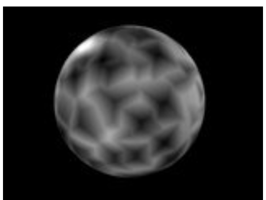
Four sliders at the bottom of the Voronoi panel represent the values of the four Worley constants (explained a bit in the Worley Documentation), which are used to calculate the distances between each cell in the texture based on the distance metric. Adjusting these values can have some interesting effects on the end result. Check the Samples Gallery for some examples of these settings and what textures they produce.

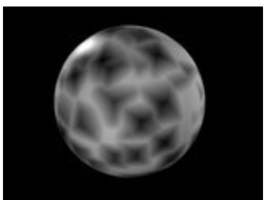
At the top of the panel there are four variation buttons which use four different noise basis as methods to calculate color and intensity of the texture output. This gives the Voronoi texture you create with the "Worley Sliders" a completely different appearance and is the equivalent of the noise basis setting found on the other textures.

Examples

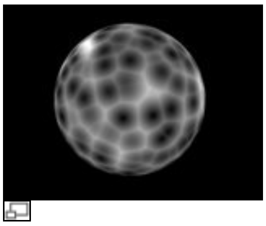
See the [Samples Gallery](#) from the release notes for more examples.





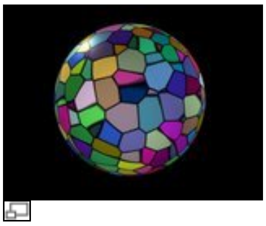


Manhattan Cebychev Minkowsky 10

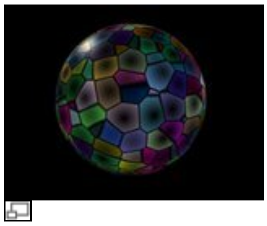


Noise Basis Int.

Noise Basis Col1.



Noise Basis Col2.



Noise Basis Col3.

Technical Details

For a more in depth description of the Worley algorithm, see: Worley Documentation.

Distorted Noise

Mode: All Modes

Panel: Shading/Texture Context Distorted Noise

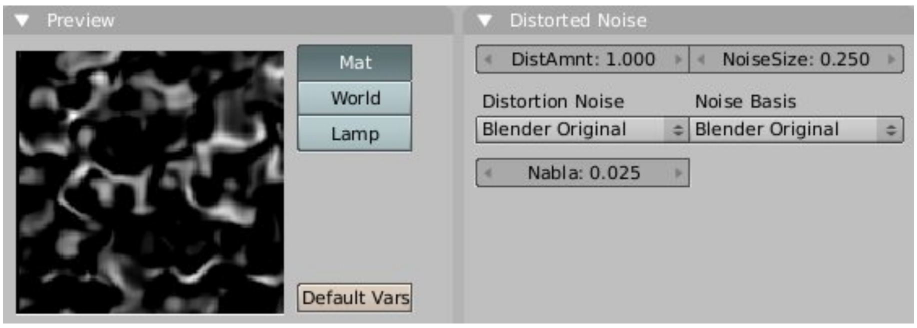
Hotkey: F6

Description

Often used for: Grunge, very complex and versatile

Result(s): Intensity

Options



Distorted Noise Texture Panels.

Distortion Noise
: The texture to use to distort another

Noise Basis
: The texture to be distorted

Noise Size
: The size of the noise generated

Distortion Amount
: The amount that Distortion Noise affects Noise Basis

Examples

See the Samples Gallery from the release notes for more examples.

Technical Details

Distortion Noise takes the option that you pick from Noise Basis and filters it, to create hybrid pattern.

Previous: Manual/Texture Options Contents *Next:* Manual/Image Textures

User Manual: Contents | Guidelines | Blender Version **2.44**

Image Textures

Mode: All Modes

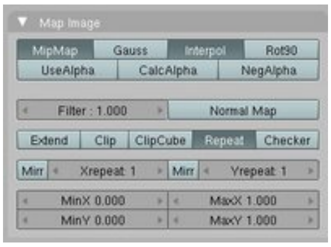
Panel: Shading/Texture Context

Hotkey: F6

Description

If you choose the Texture Type Image in the Texture Panel, the Map Image and Image panels appear, allowing you to control most aspects of image textures and how they are applied.

Map Image Panel Options



Map Image Panel in the Texture Buttons (F6).

This panel controls what aspects of the image are used, how it is to be mapped to the underlying material, and controls whether it is to be offset from its origin, and whether it should be repeated or stretched to fit.

Background image

Foreground image

These two different images are used here to demonstrate the different Map Image options. *The background image* is an ordinary JPG−file, *the foreground image* is a PNG−file with various Alpha− and Greyscale values. The vertical bar on the right side of the foreground image is an Alpha blend, the horizontal bar has 50% Alpha.}}

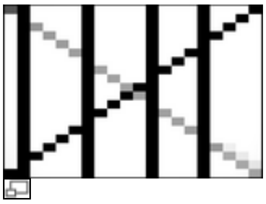
MipMap
: MipMaps are precalculated, smaller, filtered Textures for a certain size. A series of pictures is generated, each half the size of the former one. This optimizes the filtering process. By default, this option is enabled and speeds up rendering (especially useful in the game engine). When this option is OFF, you generally get a sharper image, but this can significantly increase calculation time if the filter dimension (see below) becomes large. Without MipMaps you may get varying pictures from slightly different camera angles, when the Textures become very small. This would be noticeable in an

animation.

Gauss

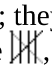
Used in conjunction with MipMap, it enables the MipMap to be made smaller based on color similarities. In the game engine, you want your textures, especially your MipMap textures to be as small as possible to increase rendering speed and framerate.

InterPol



Enlarged Imagetexture without InterPol.

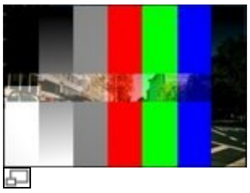
Enlarged Imagetexture with InterPol.

This option interpolates the pixels of an Image. This becomes visible when you enlarge the picture. By default, this option is on. Turn this option OFF to keep the individual pixels visible and if they are correctly anti−aliased. This last feature is useful for regular patterns, such as lines and tiles; they remain 'sharp' even when enlarged considerably. When you enlarge this 10x10 pixel Image , the difference with and without InterPol is clearly visible (*Enlarged Imagetexture*). Turn this image off if you are using digital photos to preserve crispness.

Rot90

Rotates the Image 90 degrees counterclockwise when rendered.

UseAlpha



Foreground Image with UseAlpha. The alpha values of the pixels are evaluated.

Foreground Image with CalcAlpha.

Works with PNG and TGA files since they can save transparency information (*Foreground Image with UseAlpha*). Where the alpha value in the image is less than 1.0, the object will be partially transparent and stuff behind it will show.

CalcAlpha

Calculate an alpha based on the RGB values of the Image. Black (0,0,0) is transparent, white (1,1,1) opaque (*Foreground Image with CalcAlpha*). Enable this option if the image texture is a mask. Note that mask images can use shades of gray that translate to semi−transparency, like ghosts, flames, and smoke/fog.

NegAlpha

Reverses the alpha value. Use this option if the mask image has white where you want it transparent and vice−versa.

Filter

The filter size used in rendering, and also by the options MipMap and Interpol. If you notice gray lines or outlines around the textured object, particularly where the image is transparent, turn this value down from 1.0 to 0.1 or so.

Normal Map

This tells Blender that the image is to be used to create the illusion of a bumpy surface, with each of the three RGB channels controlling how to fake a shadow from a surface irregularity. Needs specially

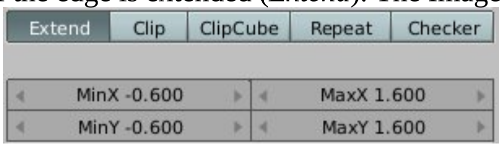
prepared input pictures. See section [Bump and Normal Maps](#).

Usually, an image texture will fit the entire size of the texture space. Similarly to the NoiseSize options in procedural textures, you can also change the scaling and positioning of image textures, within the texture itself, before it is mapped.

Extend

Extend. Outside the Image the colour of the edge is extended.

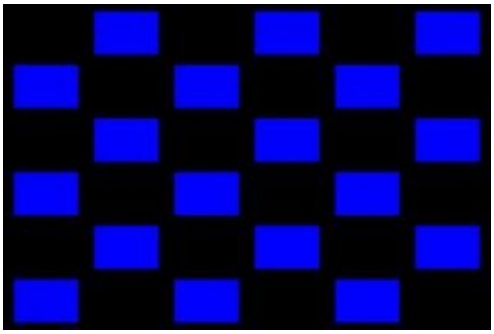
Outside the Image the colour of the edge is extended (*Extend*). The Image was put in the center of the object with the shown settings.



Clip
: Outside the Image, an alpha value of 0.0 is returned. This allows you to 'paste' a small logo on a large object.

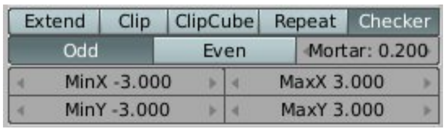
ClipCube
: The same as Clip, but now the 'Z' coordinate is calculated as well. Outside a cube−shaped area around the Image, an alpha value of 0.0 is returned.

Checker



Checker generates Checkerboard patterns. Here a blue picture was used on a black background.

Checkerboards quickly made. Mortar governs the distance between the checkers in parts of the texture

size. 

(You can use the option size on the Map Input Panel as well to create the desired number of checkers).

Mirror

The two Mirr buttons allow you to map the texture as a mirror, or automatic flip of the image, in the corresponding X and/or Y direction.

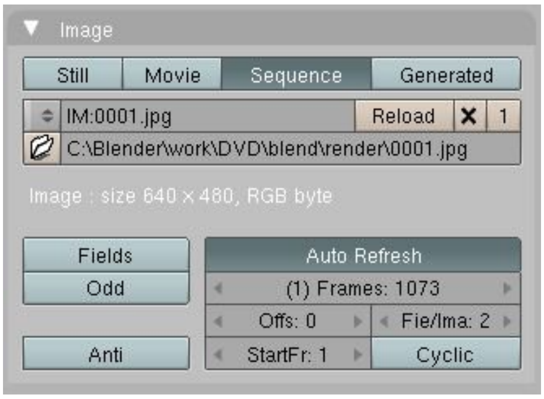
Repeat

The Image is repeated horizontally and vertically as often as set in Xrepeat and Yrepeat.

MinX, MinY, MaxX, MaxY

The offset and the size of the texture in relation to the texture space. Pixels outside this space are ignored. Use these to crop, or choose a portion of a larger image to use as the texture.

Image Panel Options



The Image Panel in the Texture Buttons (F6).

Load Image

- Load a single image file in one of Blender's supported file formats: BMP, JPG, PNG, TGA, TIFF, OpenEXR, Cineon, DPX and Radiance HDR. Others, like PSD and GIF − are partially supported via QuickTime on Windows and Mac versions.
- To use a vector format SVG image, use the [Vectrex texture plugin]
- Load an animation as a numbered image sequences in any of the supported image file formats. To do this, first click on the first frame and then Load file. Then change the Image type to Sequence, and enter the ending frame number
- Load an animation as an avi or mov file. Accepted are uncompressed or JPG compressed AVIs. On Windows and Mac with QuickTime, most quicktime movies should work, and on Windows platforms all videos with a supported codec should work.

For any of the above, you can use absolute as well as relative Paths. The // (double Backslash) means the working directory, .. (two dots) point at the parent directory. Select relative or absolute in the file browser window header.

Movie

Movie files (AVIs supported by Blender, SGI−movies) and "anim5" files can also be used for an Image texture. In this case the subsequent Panel, Anim and Movie is populated by Buttons.

IM

The internal name of the IMage. Use the select button to rapidly change between textures loaded into memory. Shift LMB click into the field to give the image a meaningful name

Reload

Load the image again from disk (i.e. if it's been changed in an external application)

X

Delete the link from this texture to this image

Users

The number of other materials that use this image. Click to make the information about this image local to this instance of use.

Pack ( )

Saves the still or generated image inside the blend−file. Use this if you are mailing or sharing .blend files between PC's with different directory structures.

The text below the filename provides information about the file

Fields

Video frames consist of two different images (fields) that are merged by horizontal line. This option makes it possible to work with field images. It ensures that when Fields are rendered the correct field of the Image is used in the correct field of the rendering. MipMapping cannot be combined with Fields.

Odd

Normally, the first field in a fielded (interlaced) video frame begins on the first line. Some frame grabbers do this differently.

Anti

Graphic images such as cartoons and pictures that consist of only a few colors with a large surface filling can be anti−aliased as a built in pre−process. This is not useful for Photos and similar pictures. Needs OSA.

Auto Refresh

When you change frames, blender gets the latest image from the sequence or codec.

Frames

How many frames of the animation to use; the length of the segment

Offs

What frame number inside the movie/sequence to start grabbing

Fie/Ima

Fields per Image. Used with Fields and interlaced video, it says whether each image has both odd and even, or just one.

StaFr

Starting Frame − when the animation reaches this frame number, the movie/sequence will start playing

Cyclic

When the video ends, it will loop around the to the start and begin playing again.

Seamless Images

The good, the bad, and the ugly

When an image is repeated, the left side of the repeat copy is mushed up against the right side. Similarly, when repeating upwards, the bottom of the second copy is placed against the top of the first. Where they meet, a seam may show if the pixels of one side do not line up with the other. You want image textures to be seamless, so that when viewed side by side, the repeating is not detectable. The best example of this is a brick texture. Use an image texture of a few bricks where the boundaries of the image align at the top and bottom on the mortar, and the left and right edges evenly split a brick, and that the left brick matches up nicely with the far right brick. This concept applies to terrain (outside grass, sidewalks, pavement), flooring (inside wood, stone, tile), tree bark, animal skins, roofing shingles of all types, cloth and fabric designs, wallpaper designs, dirty surfaces, sidewalks, and almost all exterior building surfaces (continuous, siding, shingle, brick, stone veneer). Both images to the right were tiled 2 times across and 3 times vertically.

When working with existing images to make them seamless, try to correct any skewing based on camera angle; you want an image taken "straight on", because the program will repeat the image flat onto the surface. Also pay attention to lighting; you want a flat light across the surface, because if the left side is lighter than the right, when they are placed together, there will be a noticeable difference.

Some paint programs have features to help you. For example, GIMP has a map filter to Make Seamless any image. It also has tools like Threshold that show you any lighting differences.

Also, placing a prop in front of a seamed wall really fakes out the eye from detecting a pattern. A simple table and lamp or chair for inside walls, or a plant or person or car for outside really helps deflect attention from percieving a repeating pattern.

Previous: Manual/Procedural Textures Contents *Next:* Manual/Animated Image Textures

Video Textures

Mode: All Modes

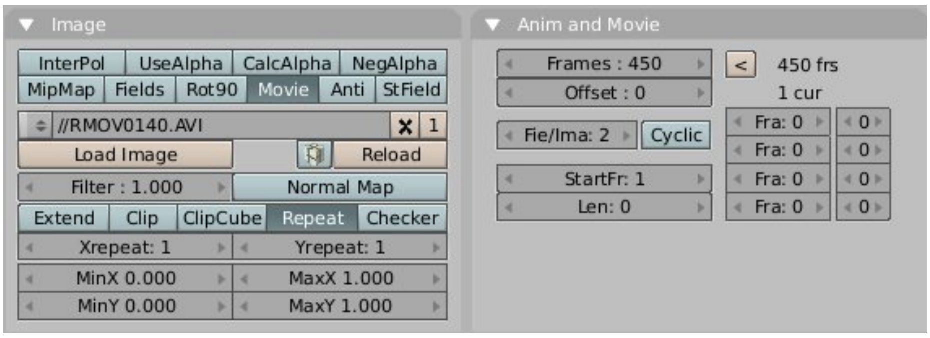
Panel: Shading/Texture Context Anim and Movie

Hotkey: F6

Description

As well as animating a texture's mapping with its associated Ipo curves (eg. the textures offset − ofsX/Y/Z), you can also use animated image sources as textures. The simplest method to get an animated texture is to use a video file. The video needs the same − or an integral division − number of frames per second (FPS) as the animation, to run at the same speed.

Options



An AVI file as an Image Texture.

frs / <
: The number frs in the Anim and Movie Panel shows how many frames were recognised. You can copy this number to the field Frames with the arrowbutton. The number cur shows, which frame is shown in the Preview Panel.

Frames
: This activates the animation option; another image file (in the same Image block) will be read per rendered frame (see also Fie/Ima). The number in the field Frames is the number of frames that shall be used in the animation. The last frame will be used as static texture for the rest of the animation, if you don't turn on the option Cyclic.

Offset
: The number of the first picture of the animation. The end frame is calculated as Frames + Offset.

Fie/Ima
: "Fields per Image:" The number of fields per rendered frame. If no fields are rendered, even numbers must be entered here. (2 fields = 1 frame). This sets the speed of the animation. The correct settings depend on the framerate of the texturevideo, the framerate of the rendered animation, whether you render Fields (in the Render Panel of the Szene context) and whether the texturevideo uses "Fields" (Fields Button in the Image Panel of the Texture buttons). Some examples:
  - ◊ The video has 24 FpS, the animation shall have 24 FpS. You are rendering without Fields. Set Fie/Ima to 2.
  - ◊ The video has 12 FpS, the animation shall have 24 FpS. You are rendering without Fields. Set Fie/Ima to 4.

◊ The video has 16 Frames, the animation shall have 96 Frames. You are rendering without Fields. Set Fie/Ima to 6.
◊ The video has 24 FpS, the animation shall have 24 FpS. You are rendering with Fields and are using Fields in the Image Panel. Set Fie/Ima to 1.

Cyclic
: The animation Image is repeated cyclically.

StartFr
: The moment − in Blender frames − at which the animation Image must start. Until that moment the first Image of the video is used as texture.

Len
: This button determines the length of the animation. A Len of 0 means, that the length is equal to Frames. By assigning Len a higher value than Frames, you can create a still at the end of the animation when you use cyclic.

Fra
: The Fra buttons allow you to create a simple montage within an animation Image. The left button, Fra indicates the frame number, the right−hand button indicates how long the frame must be displayed (Stutter−Mode). An example follows. If you use Fra you have to set Frames and Len accordingly.

Brightness

To keep the original brightness of the video, use the option **Shadeless** for the material.

Numbered Image Sequences

Instead of a video file you can also use a numbered image sequence. The simplest procedure will be to save the images in a subdirectory to your blend file and to load **one** of the images from this directory.

Examples

This example uses an image sequence instead of a video − 12 Image files (01.jpg to 12.jpg). The entry in the field Frames activates the animation. Now Blender tries to find the next frames by changing a number in the filename. You may not use the option Movie also!

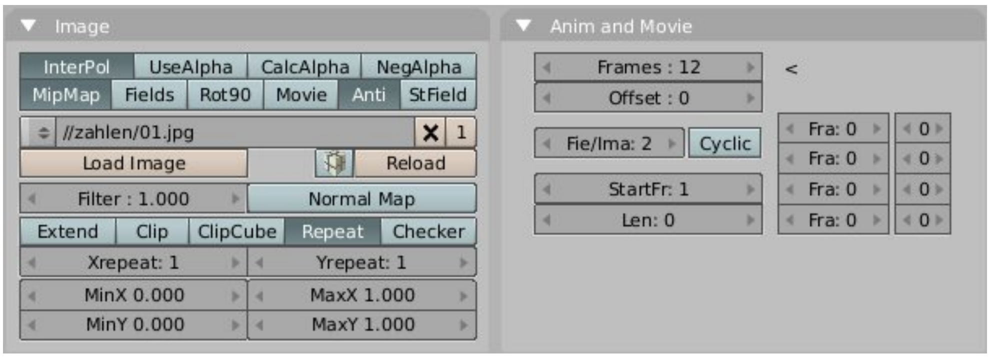
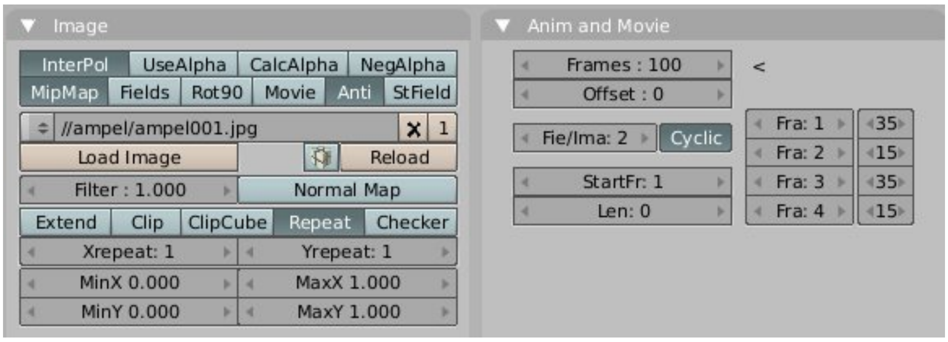
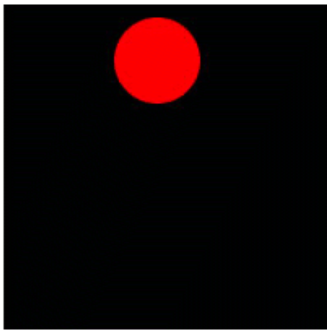


Abbildung 2: Nummerierte Bilddateien als Textur

Everything else works similarly to video textures. Here's an example of the Fra Option.
Lets assume, you would like to create an animated Traffic light. At first you create four different images: Red, Red/Yellow, Green, Yellow. These four images shall change continuously.



Settings for the animation of a Traffic light with four images. Idea by "tordat" [1]



The Traffic light with the settings from the picture *Settings for the animation*.

Please note that you need 100 Frames, though only four images are used. The result is not surprising (*The Traffic light*).

Limitations

When using image sequences, your image sequence must be named a certain way, to keep Blender counting:

1. Blender tries to find the other files by changing a number in the file name. Only the rightmost digit is interpreted for this. For example: 01.ima.099.tga + 1 becomes 01.ima.100.tga.
2. The numbers have to have the samge length. Amend leading zeros. Blender counts from 1 to 12, but not back to 1. Instead it begins at 10. So use 01, 02 ...

User Manual: Contents | Guidelines | Blender Version **2.40**

Environment Maps

Mode: All Modes

Panel: Shading/Texture Context Envmap

Hotkey: F6

Description

Environment maps take a render of the 3D scene and apply it to a texture, to use for faking reflections. If you want to achieve a very realistic result, raytraced reflections are a good solution. Environment Maps are another way to create reflective surfaces, but they are not so simple to set up.

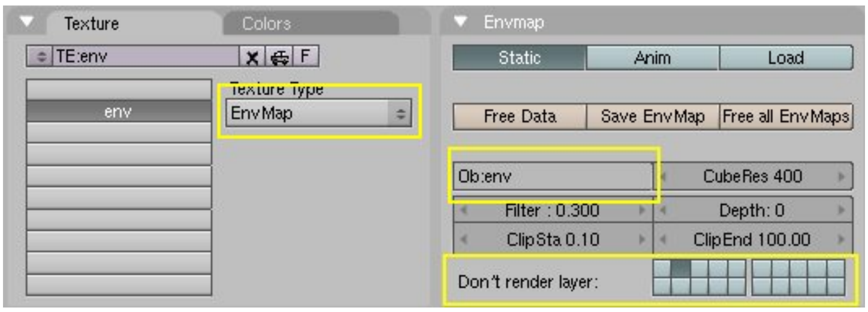
So why should one use Environment Maps? The main reason is probably that they can be much faster than raytracing reflections. In certain situations they need to be calculated only once, and may be reused like any ordinary texture. You may even modify the precalculated Environment Map in an image editor. Environment maps can also be blurred and render even faster because the resolution can then be lowered. Blurring a reflection with the raytracer always adds to the render time, sometimes quite a lot.

Just as we render the light that reaches the viewing plane using the camera to define a viewpoint, we can render the light that reaches the surface of an object (and hence, the light that might ultimately be reflected to the camera). Blender's environment mapping renders a cubic image map of the scene in the six cardinal directions from any point. When the six tiles of the image are mapped onto an object using the Refl input coordinates, they create the visual complexity that the eye expects to see from shiny reflections.

Note

It's useful to remember here that the true goal of this technique is *believability*, not *accuracy*. The eye doesn't need a physically accurate simulation of the light's travel; it just needs to be lulled into believing that the scene is real by seeing the complexity it expects. The most unbelievable thing about most rendered images is the sterility, not the inaccuracy.

Options



Reflecting plane EnvMap settings.

Important

For correct results, the mapping of an environment map texture must be set to 'Refl' (reflection co−ordinates) in the Map Input panel of the Material context.

Ob
: Environment maps are created from the perspective of a specified object. The location of this object will determine how 'correct' the reflection looks, though different locations are needed for for different reflecting surfaces. Usually, an Empty is used as this object.

  ◊ For planar reflections, the object should be in a location mirrored from the camera, on the other side of the plane of reflection (see Examples). This is the most accurate usage of Environment maps.
  ◊ For spherical reflections, the object should be in the center of the sphere. Generally, if the reflecting sphere's object center point is in the center of its vertices, you can just use the name of the actual sphere object as the Ob:
  ◊ For irregular reflections, there's no hard and fast rule, you will probably need to experiment and hope that the inaccuracy doesn't matter.

Don't render layer
: The layer to exclude from the environment map creation. Since environment maps work by rendering the scene from the location of the Ob: object, you will need to exclude the actual reflecting surface from the environment map, otherwise it will occlude other objects that should be reflected on the surface itself.

  Eg. If you are rendering an environment map from the center of a sphere, all the environment map will show by default is the inside of the sphere. You will need to move the sphere to a separate layer, then exclude that layer from the environment map render, so that the environment map will show (and hence reflect) all the objects outside the sphere.

CubeRes
: The resolution of the cubic environment map render. Higher resolutions will give a sharper texture (reflection), but will be slower to render.

Filter
: The amount of blurring applied to the texture. Higher values will blur the environment map to fake blurry reflections.

Depth
: The number of recursive environment map renders. If there are multiple reflecting objects using environment maps in the scene, some may appear solid, as they won't render each other's reflections. In order to show reflections within reflections, the environment maps need to be made multiple times, recursively, so that the effects of one environment map can be seen in another environment map. See Examples.

Clipsta/ClipEnd
: The clipping boundaries of the virtual camera when rendering the environment map

Blender allows three types of environment maps, as you can see in *Reflecting plane EnvMap settings.*:

Static
: The map is only calculated once during an animation or after loading a file.

Anim
: The map is calculated each time a rendering takes place. This means moving Objects are displayed correctly in mirroring surfaces.

Load

When saved as an image file, environment maps can be loaded from disk. This option allows the fastest rendering with environment maps, and also gives the ability to modify or use the environment map in an external application.

When using planar reflections, if the camera is the only moving object and you have a reflecting plane, the Empty must move too and you must use Anim environment map. If the reflecting object is small and the Empty is in its center, the environment map can be Static, even if the object itself rotates since the Empty does not move. If, on the other hand, the Object translates the Empty should follow it and the environment map be of Anim type.

Free Data
: Clears the currently rendered environment map from memory. This is useful to refresh a Static environment maps and you have changed things in your scene since the last time the environment map was rendered. Anim environment maps do this automatically on every render.

Save EnvMap
: Saves the currently stored static environment map to disk as an image file. This can be loaded again with Load.

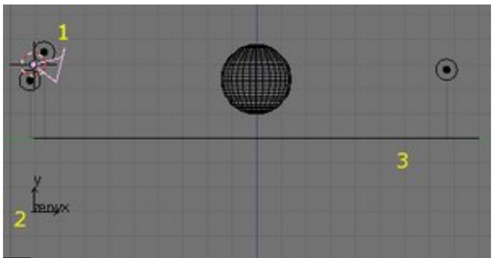
Free all EnvMaps
: Does the same as Free Data, but with all environment maps in the scene. This is a useful shortcut when using recursive environment maps (when the Depth is greater than 0).

Note

EnvMap calculation can be disabled at a global level by the EnvMap Tog Button in the Render Panel of the Rendering Buttons.

Examples

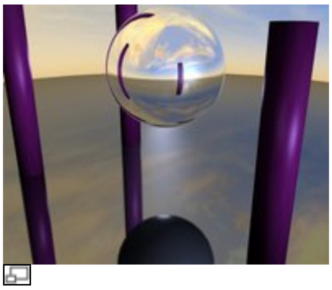
In this example, an empty is used as the Ob: of the reflecting plane's environment map. It is located in the specular position of the camera with respect to the reflecting surface. (This is possible, strictly speaking, only for planar reflecting surfaces.) Ideally, the location of the empty would mirror the location of the camera across the plane of the polygon onto which it is being mapped.



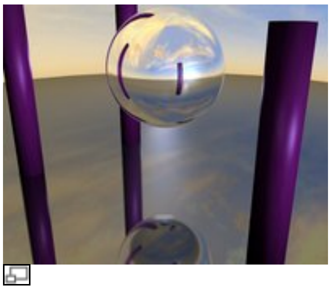
Planar reflection example. 1: Camera, 2: Empty, 3: Reflecting Plane.

Sphere on a reflecting surface.

The following images show the effect of the Depth. The first render has depth set to 0. This means the environment map on the plane has rendered before the environment map of the sphere, so the sphere's reflection isn't shown. By raising the Depth, the environment map is rendered recursively, in order to get reflections of reflections.



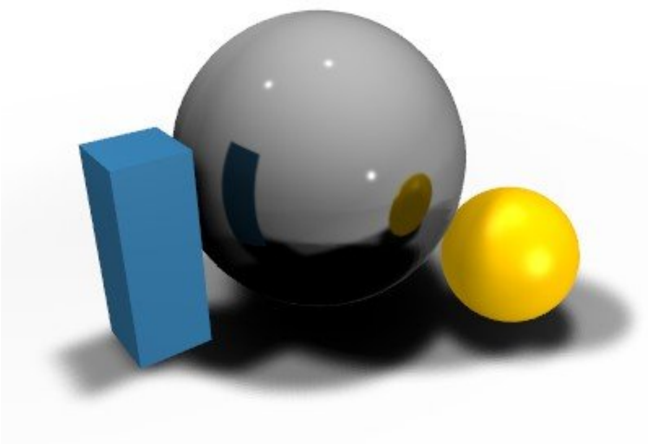
Reflecting sphere on a reflecting surface.



Reflecting sphere on a reflecting surface with multiple reflections.

Limitations

Because environment maps are calculated from the exact location of the Ob:'s object center, and not from actual reflecting surface, they can often be inaccurate, especially with spheres. In the following image, the rectangular prism and the smaller spheres are touching the sides of the large reflecting sphere, but because the environment map is calculated from the center of the sphere, the surrounding objects look artificially far away.



Inaccurate spherical reflection, the coloured objects are artificially offset

Previous: Manual/Animated Image Textures	Contents	*Next:* Manual/Texture Plugins

User Manual: Contents | Guidelines | Blender Version **2.45**

Texture Plugins

Mode: All Modes

Panel: Shading/Texture Context Plugin

Hotkey: F6

Description

Texture plugins are external files that can be loaded up in the Blender interface, which provide controls in the Plugin panel based on what they can do. A plugin texture is a dynamically loaded library that exists as a separate file on your computer. When called in, it generates the texture. A standard set of plugin textures are distributed with Blender and are located in the `Blender Foundation/Blender/plugins/texture` directory, or wherever the pathspec points to in the User Preferences â†' File Paths â†' Tex Plugins field.

These plugins are developed by various people, and a general central collection site is hosted at the [Blender Plugin Repository]. A recent brick texture can be found through the BlenderArtists forum. When you find a good texture plugin, we recommend that you put a copy in your personal library (lib/plugin/texture) directory.

V.: 2.46

Plugins now work correctly with multi−threaded rendering. Prior versions using multi−threads will result in black or white spots or streaks in the render. To work around this, use single−threaded rendering.

Sample List of Available Texture Plugins

For the current list, examples, and more information about each plugin, please click the above link. Just to give you a quick idea as to the diversity and range of textures that are available as of January 2008:

afterglow
: Afterglow unlike Other plugins is both a Texture and a Sequence plugin for glowing objects in Blender.

brick
: Brick−drawing plugin, based on a sample plugin from NaN.

brick1
: Creates a brick texture.

ceramictiles
: This is a C implementation of the BMRT ceramictiles shader written by Larry Gritz.

chip
: This code is based at Ken Perlin's famous works...

chip2
: This code is based at Ken Perlin's famous works...

circdots_rgb
: blender plugin for generating circular dots (source written/compiled by sylvio sell, 2006)

clouds2
: Creates a nice cloud texture similar to the builtin one.

dots2
: Makes nice pok−a−dots.

fixnoise
: Creates a random static noise map, use this if your doing an animation instead of the builtin noise.

greenmarble
: This is a C implementation of the BMRT greenmarble shader written by Larry Gritz.

led
: Creates LED numbers like on a digital clock.

lyapunov
: blender plugin for generating Lyapunov fat fractals. (source written/compiled by sylvio sell, dec 2004)

mandeltex
: blender plugin for exploring Benoit Mandelbrot's fractal set. (source written/compiled by rob haarsma, between dec 95 and april 1999)

matrix
: Creates noise similar to clouds and novichip.

musgrave
: a procedural texture plugin for blender that generates (and bumps !) Kent Musgrave's fractal noise patterns.

novichip
: Another noise plugin similar to clouds or matrix

pattern
: This plugin uses the sin function to create various patterns Similar to the sinus plugin a bit different however.

pie
: blender plugin for pie shaped divs.

r_weave
: Weave plugin that does fake bumpmapping and is tileable.

refract
: Fakes Refraction.

rings2
: This produces a ring−like pattern, much like the wood texture, but this is a 4D texture. (OK, 3D really, but the third is time, not Z). acceleration.

rtilings
: with this plugin you can create regular tilings of a plane (based on squares, triangles or hexagons).

sarah0
: The sarah0 plugin (a.k.a. Sarah's first plugin) If you like this plugin you should really check out the main website for the latest version. Bricks, tiles, interlocking patterns, you name it.

scales
: This plugin generated a texture pattern that resembles fish scales. Through settings, I suppose it could also do a round shingled roof.

sinus
: This plugin uses the sin function and some constants you can modify to get different textures

spirals
: Spiral Drawing plugin.

t_bricks
: This plugin generates brick textures with bumpmapping.

t_clouds
: This plugin generates cloud textures with bumpmapping.

t_marble
: This plugin generates marble and or wood textures with bumpmapping.

t_marble_terrain
: This plugin generates marble textures with slope and altitude constraints and bumpmapping.

t_terrain
: This plugin generates cloud textures with slope and altitude constraints and bumpmapping.

t_wood

This plugin generates wood textures with bumpmapping.

tiles
: Creates a nice checkerboard pattern you can also add some noise to it.

trellis
: Create various regular 2D tiling patterns, also a true 3D cubic texture (like cells in POVRAY)

voronoi
: a procedural texture plugin for blender that generates Voronoi cell structures.

water
: Animate various kinds of ripples of water, like rain or just a dripping tap. There are many settings and they can be distorted by 'wobbly' noise. Very nice for all kinds of "wet" effects.

wbricks
: more advanced brick texture

Options

Load Plugin
: Opens a file select window to browse for a plugin to load. These plugins are **dll** files on Windows, **.so** files on Mac and various Unix flavors.

Once a plugin is loaded it turns the Texture Buttons window into its own set of buttons, as described in the individual plugin references.

Technical Details

Blender allows the dynamic linking at run time of shared objects, both texture and sequence plugins. In both cases these objects are pieces of C code written according to a given standard (*chapter_plugin_reference*). In the case of texture plugins, these chunks of code define functions accepting coordinates as input and providing a Color, Normal and Intensity output, exactly as the procedural Textures do.

See Also

- Blender's Plugin System

Previous: Manual/Environment Maps	Contents	*Next:* Manual/Painting

User Manual: Contents | Guidelines | Blender Version **2.3**

The purpose of this section is to guide you in understanding the full array of Blender's shading options and how to use them. Shading is the process of applying color, textures, and finishes to your meshes in order to simulate a wide variety of appearances, including patterns, actual painting and detailing, faces of people and animals in a variety of settings.

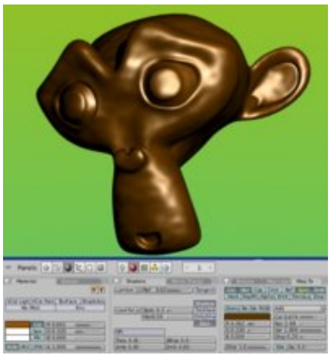
While Blender does offer painting abilities, those abilities are geared toward animation or coloring meshes. At a fine level of detail, such as skin UV textures or matte paintings, it does not compete and does not hope to compete with ease of use and functionality in specialized paint programs such as Gimp or Photoshop. Instead, it seamlessly uses their output graphic files for mesh and scene colorization.

A Review

In the previous sections, we discussed Procedural Materials and Textures. These types of materials and textures are applied as a procedure across the entire mesh. In Multiple Materials we discussed how to apply a material to a set of faces. Since a mesh can have many sets of faces, a mesh can have many procedural materials, each with its own color and other material settings, as well as procedural textures.

In Textures, we discussed how textures such as clouds or wood grain can have subtle or dramatic effects on the appearance of a mesh, and how they overlay or enhance the base material in some way (depending on how they Map Output) and mix with previous textures.

The Next Level



Chocolate Monkey by Roger

We now will take our shading to the next levels: vertex painting and then UV texturing using images. Please note that these levels of complexity and detail work together. Images can be combined with vertex colors to make the texture brighter or darker or to give it a color. Using an special texture, the Image Texture, as a UV Texture is a way of giving very detailed, precise coloring to a mesh in order to achieve photo−realistic imagery.

Consider our favorite model, Suzanne the Monkey. Below you will see, in order of increasing realism (and complexity), the results that can be achieved with each style of painting.

In the picture to the right, a single material, brown, was used to color Suzanne. A cloud texture was applied to the Color to blend in a darker brown, as well as making it look bumpy (Normal) and to vary the specularity. Use this paint technique for an object that is physically made from a single material, like chocolate, plastic, or

metal. Use one of the grain textures for wood or marble. Or a milk chocolate monkey; just eat the ears first.

Suzanne by Roger using Multiple Materials

In the picture to the right, a few materials, brown for fur, pink for nose and lips, etc. was used to color Suzanne. A cloud texture was applied to color her skin, and a veronni texture to her pink nose and eyes. A Vertex Group, Scalp, was defined to generate static particles (hair).

Mohawk Suzanne by RamboBaby using Vertex Painting

In the picture to the right, Vertex Painting was used to color Suzanne's mesh. A green scalp was used to generate a mohawk using static particles. Her right eye has the center vertex duplicated, set to single user, and a halo material was used to put "a twinkle in her eye". Her left eye has the cornea rotated to so that the vertex colors twist. Both eyes were filled in on the back side and vertex painted with green colors and also have Z−Trans enabled to allow the back to show through. Lights are placed in front of her eyes to highlight them.

Since Vertex painting shades between faces, notice how the skin colors blend more smoothly than the previous example. Her skin has a more natural appearance with colors blended (and not being blocky or single−colored).

Image:Manual−Painting−uvtexture.jpg

Suzanne by joe using UV Textures

In the picture to the right, a single image was used as a UV Texture to color Suzanne. A xxx texture was applied to xxx.

Chapters

This section tells you how to achieve these higher−quality results:

- Vertex Painting

Previous: Manual/Texture Plugins | Contents | *Next:* Manual/Vertex Paint

User Manual: Contents | Guidelines | Blender Version **2.42**

Blender features a built−in painter that allows you to paint your mesh all sorts of pretty colors. This section describes the Vertex Painter, which paints your base mesh by assigning colors to vertices, and blending those to give a face a color. Just like having your own virtual airbrush.

Setting Up

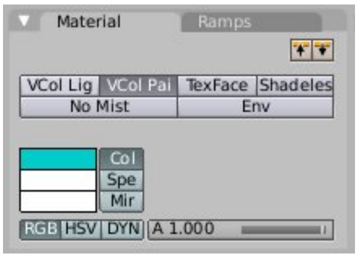
Mode: To be able to paint a mesh, you must select the object and go into Vertex Paint mode using the mode selector on the 3D Window header. Your cursor in the 3D window will switch to a cute little paintbrush.

Viewport Shading: To be able to **see** what you are painting, select Shaded as the Viewport Shading (also in the 3D window header). You don't have to be in Edit mode to paint, and although some find it distracting to see all those dots, it sometimes helps to see where the vertices actually are.

Textured Viewport Shading

If your 3D View is in Textured Viewport Shading mode, any Vertex Color paint layer will override a UV Texture layer, but will not affect rendering. In Shaded view, the correct view is shown: UV Textures override any Vertex Paint layer.

Detail: The more vertices you have, and the denser they are, the more detailed the paint effects will be. (The editing Subdivide button is really handy if you need it).



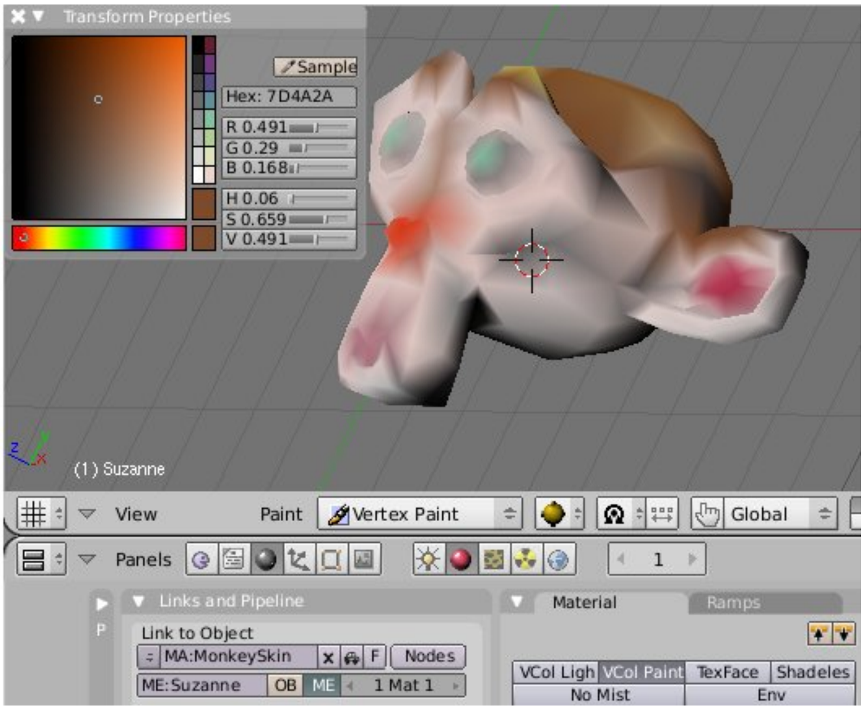
Enable Vertex Colors when Rendering

Render: You can paint all day inside Blender without having a material assigned to the object. However, you will not be able to Render without one. In the Buttons window, the Shading F5 buttons, assign a new material if the object does not have one. **Enable the VCol Paint button** in the material settings of the Shading buttons. This tells Blender to use the vertex colors you are going to paint (instead of the base material color) when it renders the image.

Multicolored lights: You can also tell Blender to use the vertex colors as a light source for the mesh when rendering; enable VCol Light and use the material Emit setting (on the Shaders panel) to vary the intensity of the light produced. Setting different colors for vertices, and then spinning the object, will make a warbling array of colors. If you want it to cast light and shadows on other objects, put a lamp inside of it and make it partially transparent (alpha less than 1.0).

Face Painting: By default, you will be painting the whole mesh. To only paint part of the mesh, press F for Faces when in Vertex Paint mode. The faces of the mesh will be outlined for you. Just like when editing a mesh, RMB clicking on a face, or Shift RMB clicking, or doing a Border select with the LMB button selects only certain faces to paint. Doing a Border select with the RMB button excludes those faces from painting. Selecting some faces and pressing H Hides them from view and thus painting (but only for the current painting session; they are unhidden when you leave paint modeâ€¦ or if you press Alt H!).

Color Picker Applet



Press traNsform to have a floating color picker applet readily available. Use the color picker by RMB clicking on a predefined color, color bar and gradient, or LMB clicking the sliders, and/or eyedropper sampler. Choose a pretty color, and LMB click−drag over your mesh to apply the color. You may supply a hissing sound when you click just to make it more entertaining.

Each time you paint a stroke without leaving the 3D window, Blender saves the stroke in the Undo buffer, so to undo a stroke, just press Ctrl Z. Set the Opacity (density) and Size of your brush in the Paint panel (in a Editing (F9) context). Many other painting controls are available in that area; see the Reference Manual for details on the painting tool.

You may move your view about in the 3D window using normal 3D space controls. Note that the window shows you what the object will look like under current lighting conditions, so just like the real world, you may have to turn the object or add/move the lights to see what your paint job really looks like.

Vertex Paint Panel

In the Editing F9 buttons, in Vertex Paint mode, there will be a Paint panel that has many controls for your paintbrush. Use the RGB sliders to choose a color the same way as the floating applet. The panel also allows you to control the paint Opacity (how 'thick' the paint is applied), the Size of the brush, and how the paint is applied to existing colors. You can apply via Mix, Add, Subtract, Multiply, Filter, Lighter, or Darker. For example, if a vertex was purple, and you set your brush to Subtract Blue, painting that vertex would make it red, since purple minus the blue is red.

To make all vertices a consistent color, set the color and click Set Vert. By default, when you paint, the paint color spreads to all faces connected to the vertice and blends in based on the size of the face. Disable All Face and Vertex to uniformly paint a face. Enabling Normals shows you the incident light appearance; use this if you have Textures applied to the material that affect the Normal.

By default, holding down the LMB is just like holding down a spray can button; the more you hold, the more paint is applied. Disable Spray and each click sprays a little bit of paint but no more, and holding down the button has no effect.

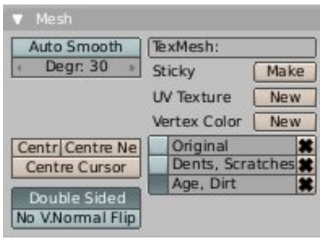
The bottom row of buttons allow you to uniformly multiply the color values, effectively increasing (a Multiply factor >1.0) or decreasing (factor <1.0). Change the value and click Set. Use Gamma correction for cross−platform image gamma correction.

Saving and Exporting

Whenever you save your .blend file, the vertex paint job is saved with it, inside the .blend file. There is nothing special you have to do.

You can also save your paint job as an external image (e.g. JPG or PNG) by baking the paint to a UV Image. You do this by unwrapping the painted mesh onto a flat surface, like carefully unwrapping a Christmas present and smoothing the paper out onto a tabletop. This process is called Baking the Vertex Paint and is [discussed in UV Texturing](#).

Overpainting and Textures



Entering Vertex Paint mode creates a Vertex Color layer, named and indicated in the Editing buttons, Mesh Panel. You can create multiple layers of Vertex Paint by clicking the New button, located in the Editing buttons, Mesh Panel, to the right of the Vertex Colors label. Each layer is painted independently, and unpainted areas of an upper layer allow more original layers (shown higher in the list) to show through. Selecting a layer shows that painting in the 3D view and/or in Render output â€“ and deactivates (hides) all those that were overlaying it! Select a layer by clicking one or both of the buttons next to its name (left one for 3D view, right one for Render), and change the name by clicking on the name and entering something creative. The example has the Original paint job of the car, overlaid with some dents and scratches, which is further overlaid by the effects of aging and dirt (a light brown−gray dusting). When rendered, all three layers will be shown, because the third layer is currently selected.

You may also paint under a UV image by enabling the use of UV images (Material Texface button), assigning faces using the UV Face Select mode, and loading the image using the UV/Image editor. Any painting you do on one object's face that is UV−mapped does not affect the UV/image, but non−mapped faces will show the vertex colors you have painted. To make permanent mods to the UV image, use the painting tool in that window via Image−>Texture Painting. Partially transparent UV images (with an alpha less than one) will allow the base vertex paint to show through.

Any texture (such as the cloud texture) that maps to the color of the material will also affect the vertex coloring.

The end color of the material also depends on the amount of ambient light it receives and the color of that ambient light. If the material is partially transparent, then the color seen will also depend on the color of the objects behind it. The ulitmate color also depends on the color of lights (lamps, reflections, radiance/glow, and other vertex color lights) that shine upon it.

Previous: Manual/Painting	Contents	*Next:* Manual/UV Unwrapping And Texturing

User Manual: Contents | Guidelines | Blender Version **2.43**

The most flexible way of mapping a 2D texture over a 3D object is a process called "UV mapping". In this process, you take your three−dimensional (X,Y & Z) mesh and unwrap it to a flat two−dimenstional (X & Y) image. Colors in the image are thus mapped to your mesh, and show up as the color of the faces of the mesh. Use UV texturing to provide realism to your objects that procedural materials and textures cannot do, and better details than Vertex Painting can provide.

UV Explained

Box being inspected

Box mapped flat

The best analogy to understanding UV mapping is cutting up a cardboard box. The box is a three−dimensional (3D) object, just like the mesh cube you add to your scene.

If you were to take a pair of scissors and cut a seam or fold of the box, you would be able to lay it flat on a tabletop. As you are looking down at the box on the table, we could say that U is the left−right direction, is V is the up−down direction. This image is thus in two dimensions (2D). We use U and V instead of the normal X and Y to avoid confusing the directions with 3D space.

When the box is reassembled, a certain UV location on the paper is transferred to an (X,Y,Z) location on the box. This is what the computer does with a 2D image in wrapping it around a 3D object.

During the UV unwrapping process, you tell Blender to map the faces of your object (in this case, a box) to a flat image in the UV/Image Editor window.

Cartography Example

Cartographers (map makers) have been dealing with this problem for millenia. A cartography (map−making)

example is creating a projection map of the whole world. In cartography, we take the surface of the earth (a sphere) and make a flat map that can be folded up into the glove compartment aboard the space shuttle. We 'fill in' spaces toward the poles, or change the outline of the map in any of serveral ways:

Mercator Projection

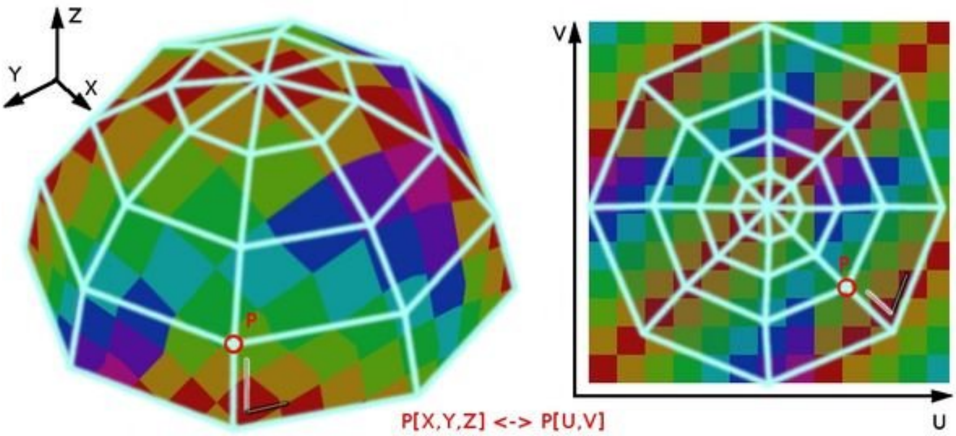
Mollweide Projection

Albers−equal Projection

Each of these are examples of ways to UV map a sphere. Each of the hundred or so commonly accepted projections has its advantages and disadvantages. Blender allows us to do the same thing any way we want to, on the computer.

On more complex models (like seen in the earth map above) there pops up an issue where the faces can't be 'cut', but instead they are stretched in order to make them flat. This helps making easier UV maps, but sometimes adds distortion to the final mapped texture.

Half−Sphere Example



3D Space (XYZ) versus UV Space (click to enlarge)

In this image you can easily see that the shape and size of the marked face in 3D space is different in UV space.

This difference is caused by the 'stretching' (technically called mapping) of the 3D part (XYZ) onto a 2D plane (i.e the UV map).

If a 3D object has a UV map, then, in addition to the 3D−coordinates X, Y, and Z, each point on the object will have corresponding U and V coordinates. (P in the image above is an example of how a point on a 3D object might be mapped onto a 2D image.)

Advantages

While procedural textures (described in the previous chapters) are useful−they never repeat themselves and always "fit" 3D objects−they are not sufficient for more complex or natural objects. For instance, the skin on a human head will never look quite right when procedurally generated. Wrinkles on a human head, or scratches on car do not occur in random places, but depend on the shape of the model and its usage. Manually−painted images, or images captured from the real world gives more control and realism. For details such as book covers, tapestry, rugs, stains, and detailed props, artists are able to control every pixel on the surface using a UV Texture.

A UV map describes what part of the texture should be attatched to each polygon in the model. Each polygon's vertex gets assigned to 2D coordinates that define which part of the image gets mapped. These 2D coordinates are called UVs (compare this to the XYZ coordinates in 3D). The operation of generating these UV maps is also called "unwrap", since it is as if the mesh were unfolded onto a 2D plane.

For most simple 3D models, Blender has an automatic set of unwrapping algorithms that you can easily apply. For more complex 3D models, regular Cubic, Cylindrical or Spherical mapping, is usually not sufficient. For even and accurate projection, use seams to guide the UV mapping. This can be used to apply textures to arbitrary and complex shapes, like human heads or animals. Often these textures are painted images, created in applications like The Gimp, Photoshop, or your favorite painting application.

Games

UV mapping is also essential in the Blender game engine, or any other game. It is the de facto standard for applying textures to models; almost any model you find in a game is UV mapped.

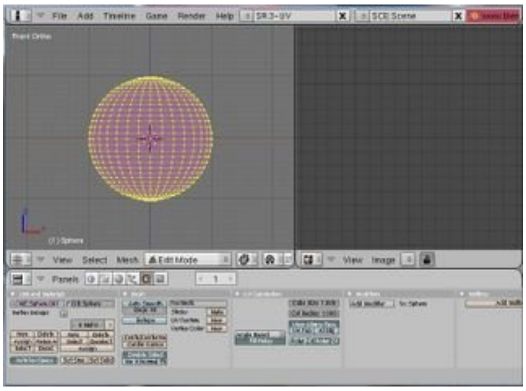
Previous: Manual/Vertex Paint	Contents	*Next:* Manual/Unwrapping a Mesh

User Manual: Contents | Guidelines | Blender Version **2.46**

The first step is to unwrap your mesh. You want to unwrap when you feel your mesh is complete with respect to the number of faces it needs to have. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you. In this fashion, you can use the UV Texture image to guide additional geometry changes.

The UV tools were significantly enhanced for Blender Version 2.46. For users of Blender 2.45 and before, consult this archived page

Getting Started

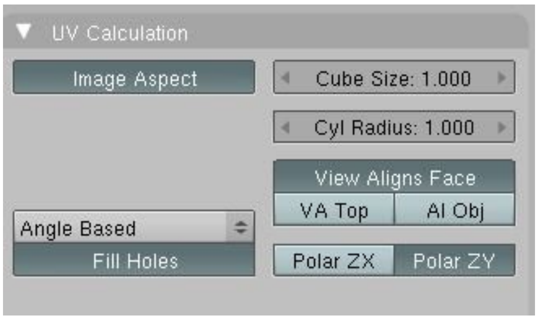


The process of unwrapping your model, called UV Unwrap, is done within Edit Mode in the 3D View window. This process creates one or more UV Islands in the UV/Image Editor window

To begin, choose the SR:3−Material screen layout from the selection list at the top of your screen in the User Preferences window header, and set one of the panes to show you the UV/Image Editor window, and another pane the 3D window.

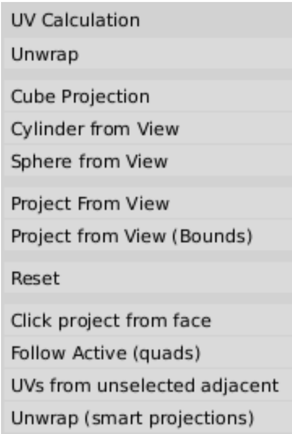
Enter edit mode, as all unwrapping is done in Edit mode. You can be in vertex, face, or edge selection mode.

Notice that the buttons window is in the F9 Editing context.



This panel controls the Unwrap process. For more information on this panel, consult the Reference manual.

Workflow



Choosing the unwrapping method

The process for unwrapping is straightforward, but there are tons of options available, each of which dramatically affect the outcome of the unwrap. By understanding the meaning behind the options, you will become more efficient at unwrapping. The process is:

1. . Set the UV Calculation panel to govern the overall unwrap process
2. . Select the object to be unwrapped in Object mode
3. . Select Textured Draw Type mode (the 3D viewport shading selector is next to the mode selector on the 3D View header)
4. . Tab into Edit mode, or select Edit Mode from the mode selector on the 3D View header
5. . Select the faces you want to unwrap. For ease of use, switch to Face Select mode by clicking the triangle icon in the Select Mode group.
6. . In the Mesh panel, activate an existing UV Texture or click the New button next to UV Texture to create a new UV Texture.
7. If you created a New UV Texture, the selected faces are automatically wrapped to Reset (explained later) and are pink if you are in Textured Draw Type mode. Un−selected faces are potato white.
8. Press U or select Mesh−>UV Unwrap. The popup menu shown to the right will appear. This menu allows you to choose what algorithm, or method, you want to use.
9. Choose the unwrap method from the popup menu
10. The method will re−calculate the location of unpinned UVs that correspond to vertices for the faces selected.

Every purple dot in the UV map corresponds to a vertex in the mesh. The dotted lines joining the UVs correspond to edges in the mesh. Each face in the UV map corresponds to a mesh face.

Each face of a mesh can have many UV Textures. Each UV Texture will have an individual image assigned to it. When you unwrap a face to a UV Texture in the UV/Image Editor, each face of the mesh is automatically assigned two extra internal features:

- *four UV coordinates* These coordinates define the way an image or a texture is mapped onto the face. These are 2D coordinates, which is why they're called UV, to distinguish them from XYZ coordinates. These coordinates can be used for rendering or for realtime OpenGL display.

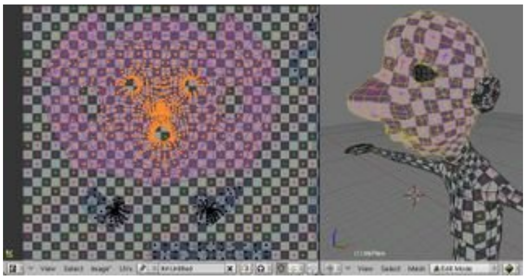
- *a link to an Image* Every face in Blender can have a link to a different image. The UV coordinates define how this image is mapped onto the face. This image then can be rendered or displayed in realtime. A 3D window has to be in "Face Select" mode to be able to assign Images or change UV coordinates of the active Mesh Object.

This allows a face to participate in many UV Textures. A face at the hairline of a character might participate in the facial UV Texture, *and* in the scalp/hair UV Texture.

Active Face

The last face clicked is "active" (either selected via right−click or de−selected thru shift−right−click). If you select some faces, then de−select some others, the last face you de−selected is "active". If the active face is a de−selected one, you will not be able to map it or see it's UV Texture, since, well, it isn't selected. All other faces that are selected are mapped though; just click on them to see their mapping.

UV Orientation



UV orientation to Face

When you map a face to a UV Texture, you have really no way of knowing the orientation. The flat square face that is a on the XY plane in 3D view could be a trapezoid on its side in the the UV/Image Editor.

The way to compare the orientation of the UV Texture face and the 3D View face is to apply a test image. In the UV/Image Editor window:

1. . Sync UV and Mesh Selection by clicking the icon on the UV/Image Editor. A little orange face−link icon shows up in the bottom left corner of the UV/Image Editor window. This shows all UV faces in the UV/Image Editor window, not just the ones selected in 3D View. This gives you a much better perspective on how all the faces are linked and laying flat in the UV/Image Editor
2. Enable View−>Draw Faces so that selected faces are pink in the UV/Image Editor which makes them easier to see.
3. . Select all the UV faces by A
4. . Create a test grid by clicking Image−>New. In the popup click UV Test Grid and click OK.

The UV Test grid is a checkerboard with a diagonal series of + in the middle in a specific vertical color rotation: magenta, purple, blue, cyan, 2 x green, yellow, orange, and then back around again. This sequence is offset by one color, so you get a diagonal checker band. This makes it very easy to see the orientation of the UV face relative to the 3D View face.

By RMB selecting the face in either window can you determine the orientation of the window, since the corresponding face in the other window will be highlighted. clicking on a face of the object. If the face was white in the 3D space, and the UV window is black with just a grid background, it means that there is no image mapped to that face. To map an image for a selected face, simply choose it from the image selector or assign a new one. and turns pink when you click, it means that there currently is no UV mapping for it.

UV Rotation

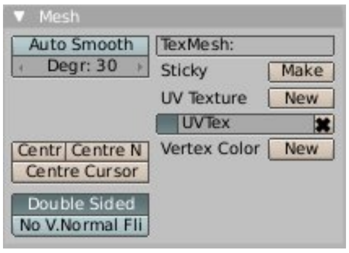
If you don't like the orientation of the UV face, for example if the picture is on its side or upside down, press R to Rotate the mapping of the selected UV's and the corresponding faces. Much like 3D View rotation, the header will show you how many degrees the mapping has rotated. Holding CTRL while rotating constrains the rotation to 5 degree increments.

UV Location

Changing the shape of a UV face (by moving the corner UVs) will also change the orientation of the mapping, since the vertex has not moved, but the mapping of image pixels to the face has (because you moved the UV location). Press G to grab the UV's and move them; LMB to drop, RMB to abort, just like vertices.

You can also snap UV's just like vertices by activating the magnet icon on the UV/Image Editor header. If you do, the Snap Target Mode selector will appear: hold control to snap UVs to the closest, center, or median of another UV island.

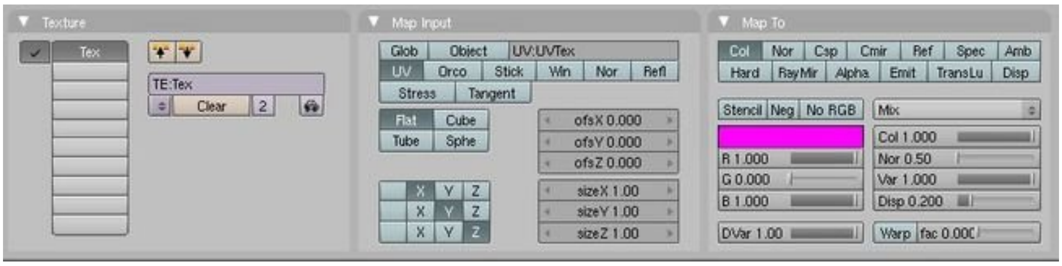
Mapping an Image to Your UVs



When you entered UV Face Select mode, Blender created a mapping in the UV/Image editor window. If you examine your Editing (F9) buttons, the Mesh panel, you will see a new UV Texture entry, called "UVTex" by default. This panel section has a button to the left of the name, used to select the UV Texture map if there are many listed. The name can be changed by clicking into the field and typing. This texture map can be deleted by clicking the X. You can add arbitrary UV Textures by clicking New, but you will rarely have to, since Blender creates them automatically each time you unwrap.

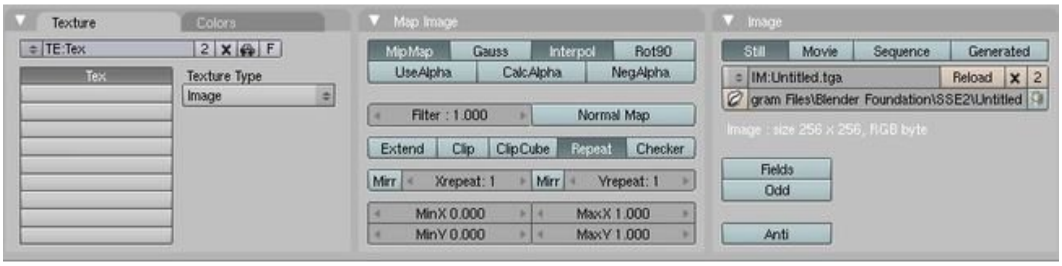
We just very briefly talked about assigning an internally generated image to this UV Layout, the Test Grid. We call that image a "UV Texture" once it is mapped to the mesh using the UV layout and applied as a texture to the material for the mesh. Let's go over that process in more detail.

First, we created the image in memory. With the faces selected in the 3D window, move on over to the UV/Image Editor window. From the header menu, select Image−>New. Here you select the size of the image to be used, and/or a test grid. Use the Test Grid to orient yourself to the direction of the UV mapping. If you select Test Grid and click OK, that image of the test grid is mapped to the faces. It cannot be rendered yet, because we have not told Blender how to reflect the light off the object when rendering. We do that by creating a Material Texture.



Material Texture settings

First, Add New material for the object. In the Material Texture tab, click Add New and a texture called "Tex" will be created and assigned to the top channel. In the Map Input panel, specify we want to use our UVTex UV−map, and map the texture to Col (which is selected by default). So, we have told Blender to color the surface of the mesh with a texture, but we haven't told Blender *which* texture to use.



Texture settings

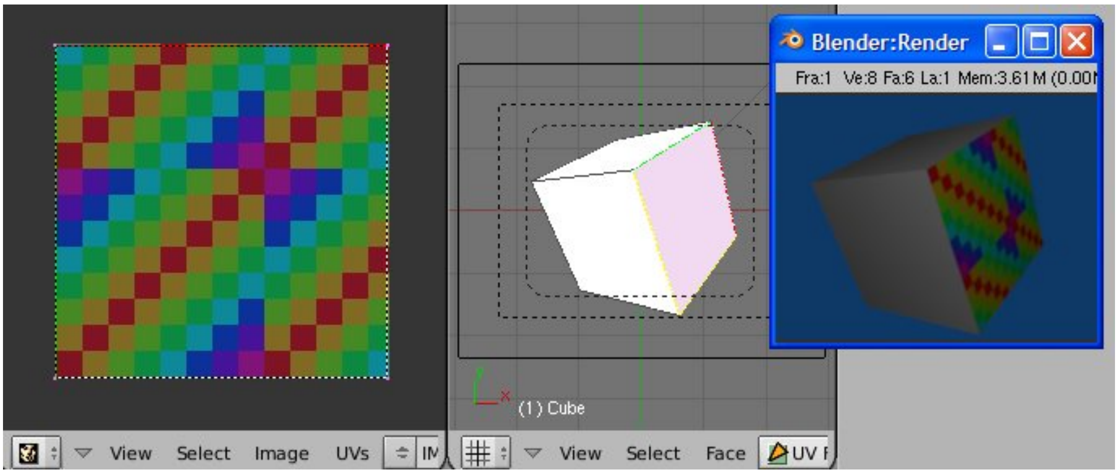
Activate the Texture subcontext buttons, and load the UV image by clicking the the Load button in the Image panel.

A quick render will show that you have automatically mapped the color of the faces of the mesh to an image used as a UV texture. Congratulations!

These three links are crucial to understanding:

1. A UV Layout, shown in the UV/Image Editor, that maps mesh faces to all or part of the image workspace,
2. A Material assigned to those faces which maps a texture to the colors, and
3. An image texture which loads an image.

Now here comes the confusing part: a mesh object can have many UV Layouts (listed in the Editing buttons panel, UV Texture list). A mesh object can have multiple materials, with a single material mapped to one or more faces of the object. A material can have many (up to 10) textures, each assigned to a texture channel. Each of those textures can be an image mapped to a UV Texture. So, Blender provides ultimate flexibility in mapping and layering images to faces of a mesh object.



Note: 3D window in Shaded viewport to show pink selected face.

You do not have to map all faces to an image. If you select the top face of the cube in UV Face Select mode, then Select−>Inverse, you will have selected all the faces except the top one. Then, in the UV/Image Editor window, select a different image. Of course, instead of creating a new image, you can load any of the image formats supported by Blender (JPG, PNG, TGA, etc.)

Once a face has been mapped, RMB clicking on that face in the 3D View will cause the UV/Image Editor window to automatically show the image and UV coordinates for the selected face. The UV map for that face will be selected in the Mesh panel.

Images versus Maps

You may have heard the terms "Bump Map" or "Spec Map". These are simply terms to denote how a UV Texture is mapped to the material, and what purpose it serves. A Color image affects the color that you see. For skin, there are four common color maps, or images, that are applied to the mesh through the UV coordinates: Tone, Spec, Blemish, and Makeup. The Tone is the basic skin color, Spec is the color you see when the light shines directly on the skin, Blemish is the wonderful imperfections: varicose veins, pimples, scars, scrapes, and makeup is coloring of the lips, eyes, cheeks. These images are *Map To Col* or color.

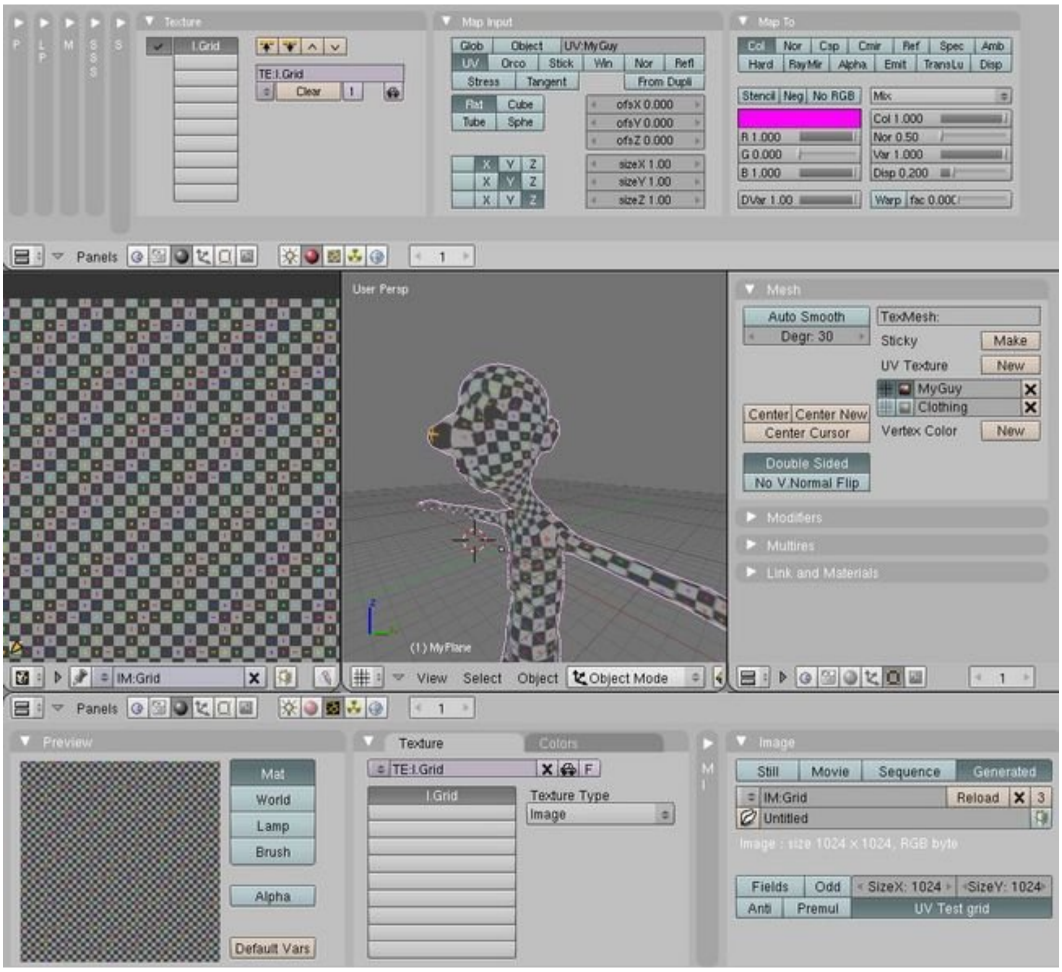
All the other images are black and white images, where black is no effect, and white is 1.0 or full effect. These images map the texture channel indicated and then on to the mesh through the UV Texture. For skin, there are commonly five: Bump (Map to Nor), Diffuse (map to Ref), Specular (map to Spec), Ambient (map to Amb), and Hardness (map to Hard). The Bump map wrinkles and gives the skin pock marks and that cellular un−eveness. The Diffuse map makes the color weaker in some areas. The Specular color indicates the skin's thickness, as thinner skin looks redder because of the blood underneath when exposed directly to light. Ambient reflects how flushed the skin is, and Hardness highlights where the skin is oily.

For example, the skin is oily around the crease of the nose. For the Hardness map then, the image would be black except where the texture maps to the area around the outiline of the nose, and there it would be white. In the Material settings, the maximum hardness might be 50. Instead of being universally hardness of 50 all over, by using the Hardness map, we can control where it is between 0 and 50.

Using UV Textures, mapped to some aspect of the texture, enables us to achieve photo realism of any surface through CG. You are not limited to these. I saw a very good zombie skin texture that used an Alpha Map to place holes in the flesh where the bone could show through (ugh).

Panel Interaction

Please excuse me for belaboring the point, but it can be very confusing. There are three different button panel sets going on here that all link to one another, as shown below.



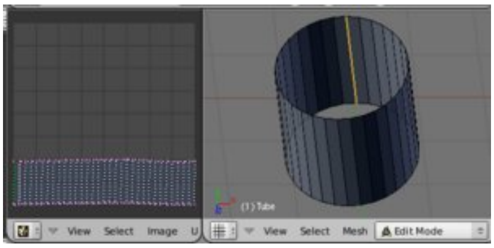
Panel Interactions

In the above example, on top is the Material buttons. The Texture panels for the Material show one texture named I.Grid, mapped to UV using the UV Texture named MyGuy. That I.Grid is mapped to affect the base color of the mesh. On the second row, you see the UV/Image Editor Window showing the Image called Grid, and the window in the center is the 3D View. The next window on the right, middle row, is the Mesh panel from the Editing context buttons, showing that we have two UV Textures; the active one called MyGuy and another one called Clothing. In the bottom row of buttons, we see the Texture subcontext of the Material buttons, and in fact see that the I.Grid texture is an Image texture that loads the Generated image that is 1K x 1K in size.

All of these names and such have to match up. I hope you can see how you can use a static image for simplicity, or use an animation to show someone blushing, or beginning to sweat, by using a movie or sequence to affect the different texture channels. I'm sure you've seen the different X−Men effects, so now you know how it was done, and how photo−realism can be achieved.

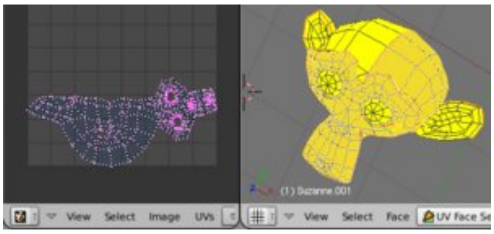
Unwrapping Using Seams

For most cases, using the Unwrap calculations of Cube, Cylinder, Sphere, or best fit will produce a good UV layout. However, for more complex meshes, especially those with lots of indentations, you may want to define a seam to limit and guide any of the unwrapping processes discussed above. Just like in sewing, a seam is where the ends of the image/cloth are sewn together. In unwrapping, the mesh is unwrapped at the seams.



Simple Seam on a Cylinder

The easiest way to define a seam is Edge selection in Edit Mode. Select the edge(s) that define the seam with border or Shift RMB, and press Ctrl E or use the Mesh−>Edges−>Mark Seam menu. In the example to the right, the back−most edge of the cylinder was selected as the seam (to hide the seam), and the default unwrap calculation was used. In the UV/Image Editor window, you can see that all the faces are nicely unwrapped, just as if you cut the seam with a scissors and spread out the fabric.

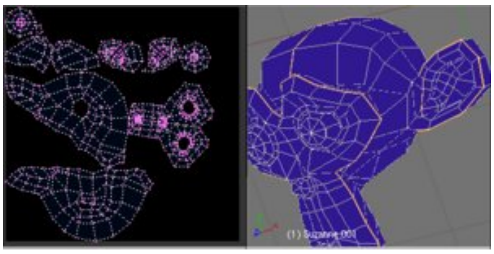


Oops! Forgot an edge in the seam

When marking seams, use the Select−>Linked Faces in UV Face Select Mode to check your work. This menu option selects all faces connected to the selected one, up to a seam. If faces outside your intended seam are selected, you know that your seam is not continuous.

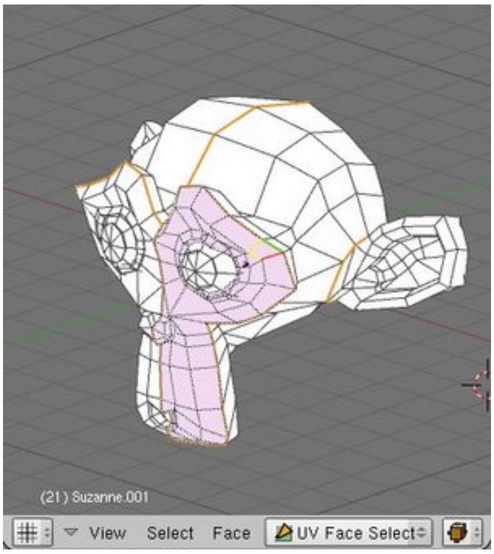
To add an edge to a seam, simply select the edge and Ctrl E Mark Seam. To take an edge out of a seam, select it, Ctrl E and Clear Seam.

You do not have to select continuous seams. If you select one edge, and follow that line and select another edge further down the line using Shift + Alt + RMB Shift ALT RMB on the edges in UV face select mode, Blender will connect the line and select all edges in between.



Seamed Suzanne

Just as there are many ways to skin a cat, there are many ways to define seams. In general though, you should think as if you were holding the object in one hand, and a pair of sharp scissors in the other, and you want to cut it apart and spread it on the table with as little tearing as possible. Note that we seamed the outside edges of her ears, to separate the front from the back. Her eyes are disconnected sub−meshes, so they are automatically unwrapped by themselves. A seam runs along the back of her head vertically, so that each side of her head is flattened out.



Face Select Mode.

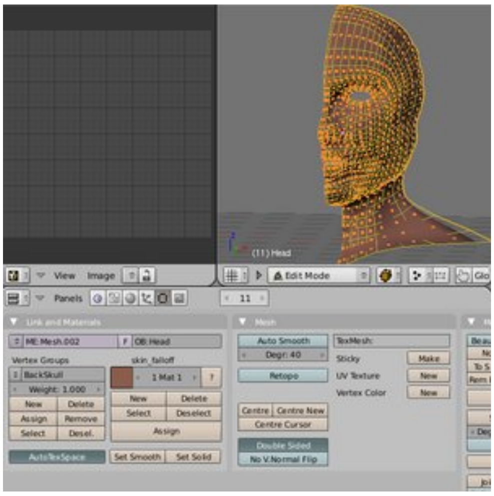
Another use for seams is to limit the faces unwrapped. For example, when texturing a head, you don't really need to texture the scalp on the top and back of the head since it will be covered in hair. So, define a seam at the hairline. Then, when you select a frontal face, and then select linked faces before unwrapping, the select will only go up to the hairline seam, and the scalp will not be unwrapped.

When unwrapping anything that is bilateral, like a head or a body, seam it along the mirror axis. For example, cleave a head or a whole body right down the middle in front view. When you unwrap, you will be able to overlay both halves onto the same texture space, so that the image pixels for the right hand will be shared with the left; the right side of the face will match the left, etc.

Unwrapping Multiple Faces

Starting Off in Object Mode

In general, you should only unwrap the faces you need to, and do so in a single unwrap operation. You only need to unwrap faces that will be painted using an image; all other faces can use procedureal materials and textures or vertex paint. You want to keep your image as small as possible, so that means you want to keep the number of faces as small as possible. For example, if the body is always going to be covered in clothes or armor, there is no need to unwrap it. If the back of the head is always going to be covered by hair, there is no need to unwrap the scalp. If you are modeling a chair with an embroidered seat cushion, you only need to unwrap the cushion and not the chair legs. In the example to the right, we only need to unwrap one side of the face, cutting our image size in half, so we leave mirror modifier on; we also do not need to double the number of UV coordinates by applying the Subsurf modifier; we can just leave it as is. Note that at this point, there is no UV Texture in the Mesh panel.



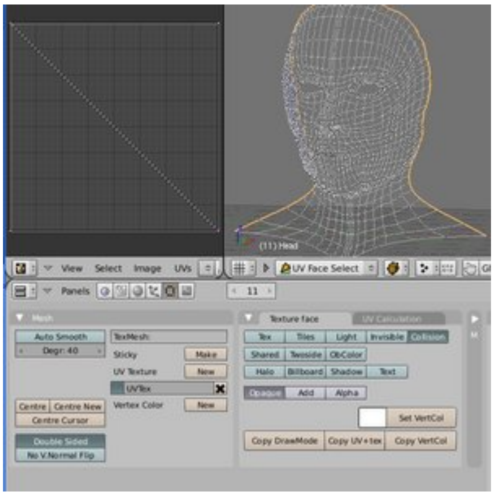
Selecting Faces in Edit Mode

To unwrap multiple faces to a single UV Texture, there are two ways to select the faces you want:

- In Edit Mode (tab), enter Face Select mode and select the faces you want.
- In UV Face Select mode using the 3D window, right−click or border select the faces you want.

The example to the right shows that we have hidden many faces from view; the ears and the back of the head. We did so by creating and using the Vertex Groups, selecting the "BackSkull" group and Hiding them from

view. We did this because we do not want to unwrap those areas, and we don't want them to get in the way during any further face selection that we may do.



Selecting Faces in UV Face Select Mode

Once we enter UV Face Select mode, Blender creates a UV Texture map for us, shown in the Mesh panel, and displays that map in the UV/Image Editor Window. By default, the map is *full−size reset overlay*, which means that each selected face is mapped to the **full size** of the UV window, and the faces are **overlaid** on top of one another. This situation is identical to the above introduction where you mapped your first face of the cube; any face of the cube you clicked on would show this same full−size map; any image selected would be displayed on **each** face. If you were to assign an image now, you would see a patchwork style of render, much like the scene in the Matrix when Neo was in the TV room and saw himself tiled over and over. If we were to use an image that was tileable, the surface would be covered in a smooth repetition of that image, with the image skewed to fit the shape of each individual face. Use this unwrapping option to reset the map and undo any unwrapping (go back to the start).

If you did not select all desired faces prior to entering UV Face Select mode, you can select multiple additional faces in any of the following ways:

- Press A and all faces of the Mesh will be selected and highlighted by dotted lines.
- Select Linked faces from the 3D View menu; this will select additional faces up to a seam
- You can select many faces by Shift RMB clicking on each one you want. You may spin the object or your view of the object and resume selecting by Shift RMB.
- BorderSelect in the 3D window.
- Enter EditMode and select the vertices or edges that define the face you want. After leaving EditMode and back in UV FAce Select mode, the faces defined by the selected vertices/edges will also be selected.

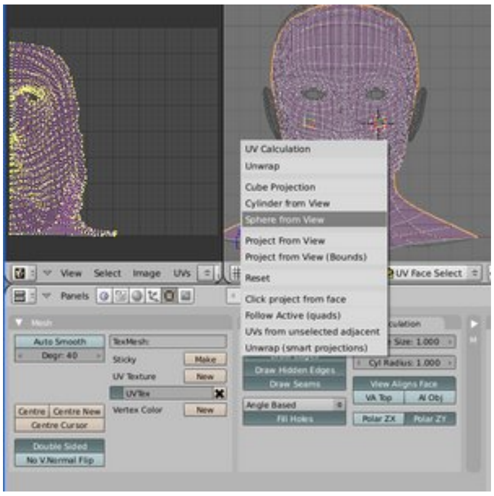
In UV Face Select mode, just like in Edit mode, you can also hide faces from view so they don't get in the way as you are trying to select faces:

- H to hide selected faces
- Shift H to hide the faces which are *not* selected, sort of the reverse of H by itself.
- Alt H to reveal the hidden faces. Note that if you leave and then re−enter UV Face Select mode, they will be unhidden.

Only one face is considered *active*; the last one selected. Since the UV/Image Editor can only show one image at a time, it will show the image for the active face. IF you change the active face mapping (if you move the

its UV coordinates), you may be overlapping or affect **all** other face mappings that are part of tha UV map (even if you cannot see them). So, before adjusting a face's UV coordinates, it is best to select adjoining faces to you see the whole map.

Face Unwrap using UV Calculation

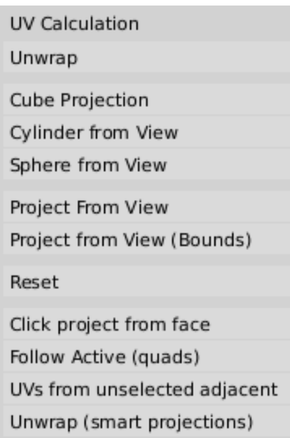


Unwrapping Faces using 3D View Menu

With our faces selected, it is now time to unwrap them to something more useful than Full−Size Reset. We want to unwrap them to the UV/Image Editor window using one of the UV Calculation shown to the right. In the 3D View header menu, select Face−>Unwrap UVs or just press U. You can control the way the faces are mapped in two ways:

- Automatically using the 3D View window command Face−>Unwrap UVs U
- Automatically using the UV/Image Editor window command UVs−>Unwrap command E
- Creating seams and then unwrapping. A seam is marked in Edit mode by selecting edges that make the seam and then issuing the command to Mark Seam.

Above, when you selected a single face and went into Face Select Mode, Blender automatically mapped the selected face to the entire image for you, based on the face's UV orientation (the red and green borders). You can unwrap a face manually through the Face−>Unwrap menu. How the selected faces map over to the image depends on the UV Calculation method that you choose.



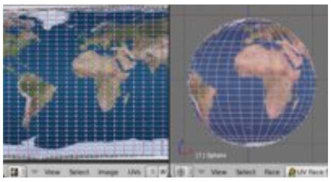
Face Unwrap Menu

The Face−>Unwrap−>Unwrap option unwraps the faces of the object to provide the 'best fit' scenario based on how the faces are connected and will fit within the image, and takes into account any seams within the selected faces. If possible, each selected face gets its own different area of the image and is not *tucked under* any other faces. If all faces of an object are selected, then each face is mapped to some portion of the image.

This point is crucial to understanding mapping later on: a face's UV image texture only has to use *part* of the image, not the *whole* image. Also, portions of the same image can be shared by multiple faces. A face can be mapped to less and less of the total image.

Based on the fundamental geometry of the object, and how it is viewed, the Face−>Unwrap−>Cube, Cylinder, and Sphere UV Calculations attempt to unfold the faces for you as an initial best fit. Here, the view from the 3D window is especially important. Also, the settings for cube size or cyclinder radius (Editing buttons, UV Calculation panel) must be set (in blender units) to encompass the object.

Normally, to unwrap a cylinder (tube) as if you slit it lengthwise and folded it flat, Blender wants the view to be vertical, with the tube standing 'up'. Different views will project the tube onto the UV map differently, skewing the image used. Also note the Polar coordinate axis selection buttons in the UV Calculation panel located in the Editing buttons window; they tell Blender which way is up.



Using a Mercator image with a Sphere Unwrap

Recall the opening cartographer's approaching to mapping the world? Well, you can achieve the same here when unwrapping a sphere from different perspectives. Normally, to unwrap a sphere, view the sphere with the poles at the top and bottom. After unwrapping, Blender will give you a mercator projection; the point at the equator facing you will be in the middle of the image. A polar view will give a very different but common projection map. Using a mercator projection map of the earth as the UV image will give a very nice planet mapping onto the sphere.

In the 3D window, Face−>Unwrap UVs−>Project from View option maps the face as seen through the view of the 3D window it was selected from. It is almost like you had x−ray vision or squashed the mesh flat as a pancake onto the UV map. Use this option if you are using a picture of a real object as a UV Texture for an object that you have modeled. You will get some stretching in areas where the model receeds away from you.

In the 3D window, Face−>Unwrap−>Reset maps each selected face to the same area of the image, as previously discussed. To map all the faces of an object (a cube for example) to the same image, select all the faces of the cube, and unwrap them using the Reset menu option.

Note: The next four options create a UV map that stretches to maximum, and does not respect the image size. After using these options, you may have to zoom waaaaay out in the UV/Image Editor, select all UV coordinates, and scale them waaaaay down to fit inside the image area.

The Face−>Unwrap−>Click project from face is a three−step process that lets you orient the projection map according to a selected face's orientation. When selected, the upper User Information bar changes to display the step you are on, temporarily replacing the Blender version and statistics. The first click selects a bottom−left corner of that typical face. The second defines 'up' based on the face orientation, so click on the corner of that face that corresponds to the up direction from the first corner. The third click defines the V coordinate, namely which way is right on the map.

The Face−>Unwrap−>Follow Active (quads) takes the selected faces and lays them out perfectly square, even if the mesh face is irregularly shaped. Note that it does not respect the image size, so you may have to scale them all down a bit to fit the image area.

Use Face−>Unwrap−>UVs from unselected adjacent to expand your UV map by selecting un−mapped (new) faces and then this option. Blender will attempt to match up the coordinates of the new face with existing faces previously mapped. This option is helpful if you have added new faces and want to unwrap them.

Finally, the Face−>Unwrap−>Unwrap (smart projections), (which used to be called the Archimapper) gives you fine control over how automatic seams should be created, based on angular changes in your mesh. This option is especially well suited to unwrapping multiple mechanical objects all at once. The Archimapper is discussed a little later on.

Face Unwrap using the UV/Image Editor Menu

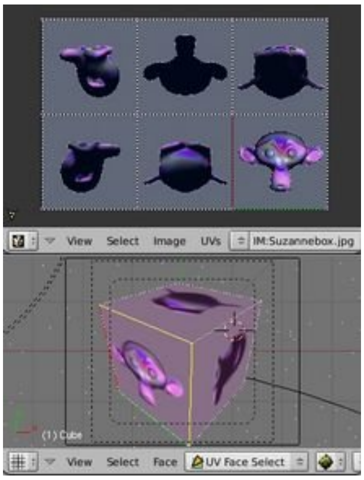
In the UV/Image Editor window menu, the menu item UVs also has a simple Unwrap option. There is no choice of different calculations to use; it is the same as selecting Face−>Unwrap UVs−>Unwrap in the 3D View. Use this option to unwrap additional faces with an existing set of unwrapped faces. The UV Coordinates for the addtional face will be added to the existing set in a non−overlapping manner. Warning: the UV coordinates for all of the faces will be recalculated.

Texture Face

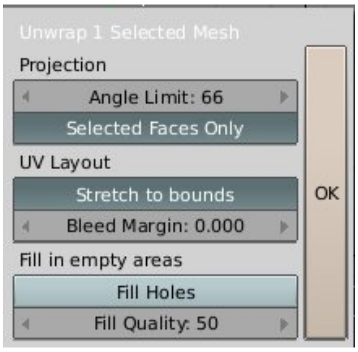
Texture Face Panel

After you have created a UV Map, an additional panel will be displayed in Edit Mode that allows you control over how the image texture is applied to the faces. Consult the Reference manual for more information on this panel.

Unwrap (smart projections)



Archimap Cube projection



Archimap control panel

An architype mapper (archimap) is the final but possibly the best option for unwrapping a mesh, even simple geometric forms. This function examines the shape of your object, the faces selected and their relation to one another, and creates a UV map based on this information and settings that you supply. In prior versions of Blender (pre−2.43), it was a script.

In the example to the right, the Archimapper mapped all of the faces of a cube to a neat arrangement of 3 sides on top, 3 sides on the bottom, for all six sides of the cube to fit squarely, just like the faces of the cube.

For more complex mechanical objects, this script can very quickly and easily create a very logical and straightforward UV layout for you.

Archimap allows you to unwrap multiple meshes in a single step (select multiple meshes in object mode). If you only have one mesh selected and are in Face Select mode, you can unwrap only selected faces, or if you select all the faces of the mesh, the whole object.

The popup panel allows the fine control over how the mesh is unwrapped. The angle limit controls how faces are grouped: a higher limit will lead to many small groups but less distortion, while a lower limit will create less groups at the expense of more distortion.

Archimap prefers you to have an image loaded if you want to Stretch to Bounds. When you use an image for a UV Texture, sometimes the edges of the image won't align perfectly or there will may be a border around the outside of the image. The Bleed Margin will shrink the overall UV coordinate map so that this margin is not mapped or used as UV Texture. For complex meshes, Archimap may create (by default) a map that has holes in it, resulting in wasted space in the image. Selecting Fill Holes will cause Archimap to do a little more work

at ensuring each face is mapped as large as possible and that there are fewest holes (wasted image texture pixels) possible.

Previous: Manual/UV Unwrapping And Texturing	Contents	*Next:* Manual/Working with UV

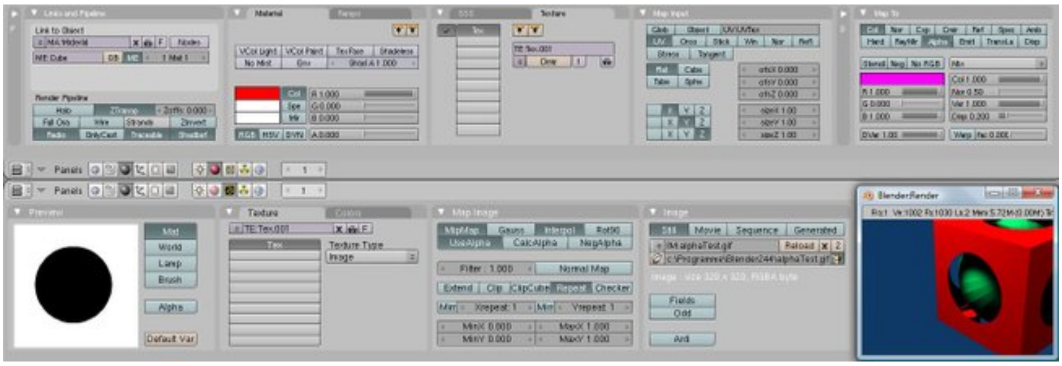
User Manual: Contents | Guidelines | Blender Version **2.46**

The second step is to work with the UV layouts that you have created through the unwrap process. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you. In this fashion, you can use the UV Texture image to guide additional geometry changes.

The UV tools were significantly enhanced for Version 2.46. For users of Blender 2.45 and before, consult this archived page

Activating UV Textures

The UV/Image Editor allows you to map textures directly to the mesh faces. The 3D View window shows you the object being textured. If you set this window into Textured viewport shading, you will immediately see any changes made in the UV/Image Editor window in this window, and vice versa.



The Material and Texture panels using a UV Texture.

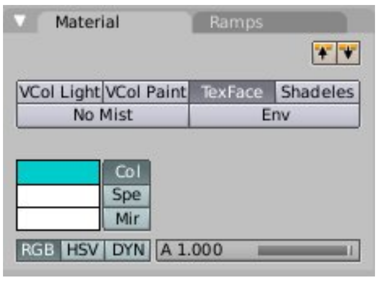
You can edit and load images, and even play a game in the Blender Game Engine with UV textures for characters and object, without a material, and still see them in the 3D window. This is because no 'real' remdering is taking place; it is all just diffuse shading.

To render an image however, you must

1. create a Material for the object, and
2. tell Blender to use the UV Textures on faces when rendering.

To create a Material, you have to click Add New Material in the F5 Shading context.

There are two ways to tell Blender to use the UV Texture when rendering: the Proper way and the Quick Way.



The Material panel with activated TexFace button.

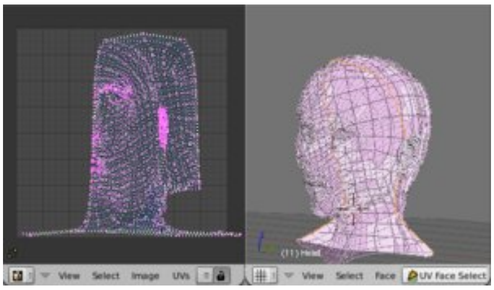
- Quick Way: The quick way is to set up a TexFace Material as shown. To do so, with the buttons window displayed, press F5 to display the Shader Buttons. In the Buttons window, Material settings, click ADD NEW material. On the Material panel, enable TexFace. This way is quick, but bypasses the normal rendering system for fast results, but results which do not respect transparency and proper shading.

- Proper way: In the Texture channel panel F6((shown above), Add a New Texture which is mapped to the UV texture (it will be mapped to Color by default, and the UV Texture is named "UVTex" by default). Select the Textures subcontext, and define the texture as an image and load the image you want to use. If the image has an alpha channel and you want to use it, click "UseAlpha" in the Map Image panel.

Material is Required for Rendering

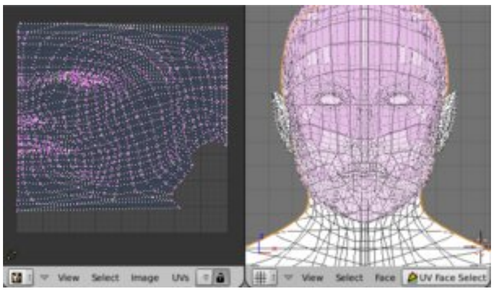
You can perform UV Texturing on a mesh within Blender without assigning a material, and you will even see it in your 3D View in textured viewport mode. However, when you render, you will just get a default gray if the object does not have a Material assigned. You will get a black if you do not load an image. If you do not create a texture that uses the image, or enable TexFace, your object will render according to the procedural material settings.

Combining UV Maps



Bad Unwrap−Note Ear and Neck

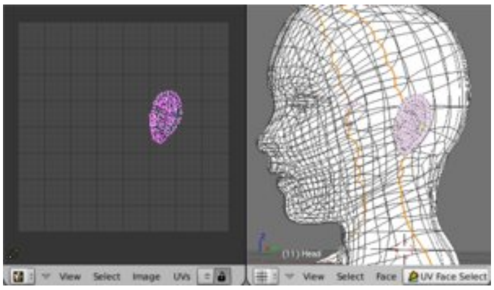
Very often you will unwrap an object, such as the face example we have been using, and get it 'mostly right' but with parts of the mesh that did not unwrap properly, or are horribly confusing. The picture to the right shows an initial unwrap of the face using the Unwrap from sphere option. The issues are with the ear; it is just a mush of UVs, and the neck, it is stretched and folded under. Too much work to clean up.



Unwrap Face Only, without Ear or Neck

We can tell that the ear would unwrap nicely with just a straightforward projection from the side view, and the neck with a tubular unwrap. So, we have to unwrap part of an object using one unwrap calculation, and the

rest using another calculation. We select only the face faces, unwrap them using the Sphere calculation, and scale and rotate them somewhat to fit logically within the image area of the UV/Image Editor window pan.

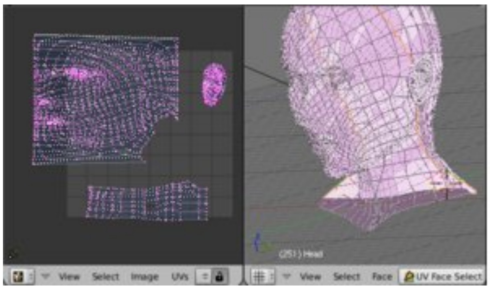


Unwrap Projection: Ear

For the next part of the mesh, unselect the faces you were working with. Their UVs will disappear, but they are still there, just not show. To verify this, you can select a few faces in 3D view and it will show up in the UV/Image Editor.

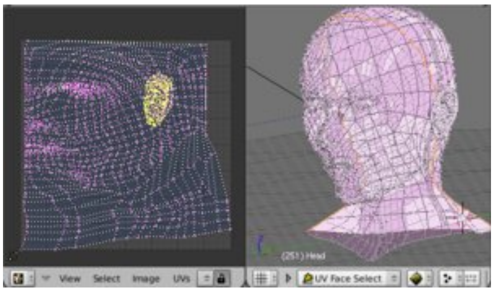
To work on the ear, in the 3D View, we select only the ear faces. You can switch out of UV Face Select mode, into Edit mode, and use Vertex Groups to select the ear. Selecting sub−meshes is easy too, since they are not connected to the rest of the mesh, simply selecting Linked vertices will select that entire submesh.

Back in UV Face Select mode, re−unwrap the ear using the Project calculation, scale and rotate them somewhat (discussed in the next section), and place them off to the side. You can do this repetitively, using different UV calculations; each re−calculation just puts those UVs somewhere else. Choose the calculation that gives you the best fit and most logical layout for subsequent painting.



UV Maps together

When all parts of the mesh have been unwrapped using the various calculations, you should end up with something that looks like to the Example to the right. All of the sections of the mesh have been mapped, and all those maps are laid out in the same UV Texture map. Congratulations! From here, it is a simple matter of stitching (discussed in the next section) to construct the entire UV Map as a single map.



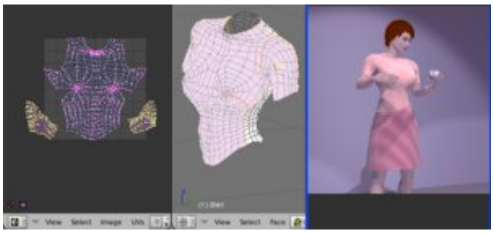
UV Maps Arranged and Stitched

When you have completed arranging and stitching, you will end up with a consolidated UV Map, like that shown to the right, arranged such that a single image will cover, or paint, all of the mesh that needs detailed painting. All of the detailed instructions on how to do this are contained in the next section. The point of this paragraph is to show you the ultimate goal. Note that the mesh shown is Mirrored along the Z axis, so the right side of the face is virtual; it is an exact copy of the right, so only one set of UVs actually exist. If more realism is desired, the Mirror modifier would be applied, resulting in a physical mirror and a complete head. You could then make both side physically different by editing one side and not the other. Unwrapping would produce a full set of UVs (for each side) and painting could thus be different for each side of the face, which is more realistic.

Averaging UV Islands

Update, added UV Island average tool, (Ctrl+A in the uv window) This makes selected UV islands proportionally equal, using their area in the 3d view

Multiple UV Layouts



Mesh with Multiple UV Textures

You are not limited to one UV Layout per mesh. You can have multiple UV layouts for parts of the mesh by creating new UV Textures. The first UV Texture is created for you when you select a face in UV Face Select mode. You can manually create more UV Textures by clicking the New button next to "UV Texture" on the Mesh panel in the Buttons Window, Editing Context) and unwrapping a different part of the mesh. Those faces will then go with that UV Texture, while the previously unwrapped faces will still go with the previous UV Texture. Note that if you unwrap the same face twice or more times (each time to a different UV Texture), the coloring for that face will be the alpha combination of the layers of those UV Textures.

In the example to the right, we have a mesh for a blouse. The mesh has been seamed as a normal blouse would, shown in the middle in UV Face Select mode. Wishing to make a cut pattern, the front of the blouse was unwrapped and basic rotation and scaling was done to center it in the UV/Image Editor window. It was

then moved off to the side, while the left and right sleeves were unwrapped, each rotated and scaled. Then, select a sample face from each cloth piece, in the 3D View Select−>Linked Faces, and the UV/Image Editor will show all those pieces (as shown to the right). You can then work with all pieces for that UV Texture layout. The example shows all three pieces moved onto the image area for painting. As you can see, the pattern nicely fits a square yard of cloth.

Another UV Layout was created by clicking the New button in the Mesh panel, and the process was repeated for the backs of the sleeves and the back of the blouse. Two images, one for the front and one for the back, are used to color the fabric. In this case, some faces map to the first texture, while other faces map to the second texture.

Additional Options

When you switch your 3D window to UV Face Select mode, Blender changes its menus slightly to give you more and better tools to use.

Similar to Active
: The Select menu now offers a script to help you select items that are similar to the last−selected (active) face. You can pick all the faces, for example, that have the same surface area (plus or minus a limit) as the active one. For each choice, you can add or take away the similar items from the list of faces selected. Criteria include Material, UV Image, Face mode, Vertex colors, and many others. For example, if you were going to change an image used as a UV texture, you would want to select all faces that used that same UV image in order to evaluate the impact of the change on the overall mesh.

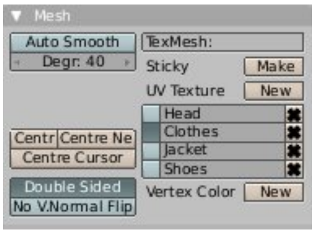
Linked Faces
: Recall that not all faces and vertices of a mesh have to be connected. A single mesh can be composed of many disconnected sub−meshes. Selecting Select−>Linked Faces is useful for selecting all the faces of a submesh. Also, linking goes up to a seam as previously discussed.

Set Vertex Colors
: If you don't like potato white, select Set Vertex Colors (Face menu) to view your mesh using the active color of Vertex Paint. A blue face color contrasts the seam colors nicely.

Editing Buttons

In the Buttons window, press F9 to get the Editing buttons. When in UV Face Select mode, the buttons change to give you additional options when working with UV maps. The AutoTexSpace button should be enabled.

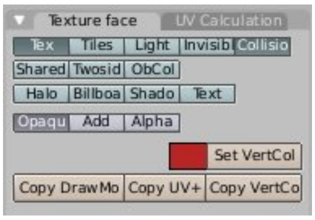
UV Textures List



The Mesh panel (shown to the right) clearly lists the UV Texture maps created for this mesh, and allows you to create New ones as placeholders for future unwrapping operations. Each map has a selector button, a name, and a fat X Delete button. The selected map is displayed in the UV/Image Editor window. The example shows a few UV maps created for a character, and the map for Clothes is selected.

Deleting a UV layout for the mesh destroys all work done in all unwrapping associated the mesh. Click with care. You've been warned.

UV Texture Options



The Texture Face panel is added when you go into UV Face Select mode, so you are not used to seeing it. Lots of neat options for controlling how the UV texture interacts with the rest of your scene. Options for how and if it is shown include: Tex, Tiles, Light, and Invisible. You almost always want it to participate in object and particle Collisions. The texture can be Shared, rendered front and back (Two Sided), and, where blank, can be shown using the object (not the vertex painted) colors.

The texture can color Halos, or be shown like a Billboard, or normally with neither of these selected. The face can used to cast Shadows on other objects.

When rendered, Opaque lets you see this texture, which is almost always a good idea. If you want sections to be transparent where the texture is, use Add; to use the texture as an alpha mask, use Alpha.

Copy UV+ is an action button, not a setting. It copies the UV mapping from the active face (the last face selected) to all other faces that are selected. Select all faces you want similarly mapped first, then Shift RMB the model face, and click Copy UV+

Drawing options (how the UV mapping looks to you while working in Blender) include drawing Faces, Edges, Hidden Edges, and Seams. When you project from a view, normally the View Aligns Face, but you can specify that, for unwrapping round objects (tubes and spheres), the view is aligned to the top (poles) of the object.

By default, Unwrap uses the Angle−Based Formula, or ABF. If you want to use Least−Squares Conformal Method (LSCM) instead, change Angle−Based to Conformal. The reason why ABF is default is because it is generally better than LSCM. There may be some exceptions for that though, so if an unwrap just isn't clean to you, try conformal.

Please set the cube and tube sizings to encompass your object prior to unwrapping, or incredible skewing and sizing will result.

Saving your UV Layout

The UV coordinates and image links are automatically saved with the mesh in the .blend file. There is nothing extra you need to do.

Saving an outline of your UV Layout

As a way of communicating to an artist who is painting your UV Texture for you, Blender offers a script called Save UV Face Layout (located in the UV/Image Editor Window, UVs−>Scripts−>Save UV Face Layout) that saves an image in Targa format (.tga) for the object you have selected. The image is an outline of the UV face mapping. The file is (by default) named with the prefix you enter. If Object name is enabled, then the name of the object is appended on so that you don't accidentally overwrite one layout for one object with another's.

Controls allow you to:

Size
: select the size of the image in pixels. The image is always square

Wire
: the thickness of the wire lines that define each face border

All Faces
: if disabled, then only the UV faces selected will be outlined

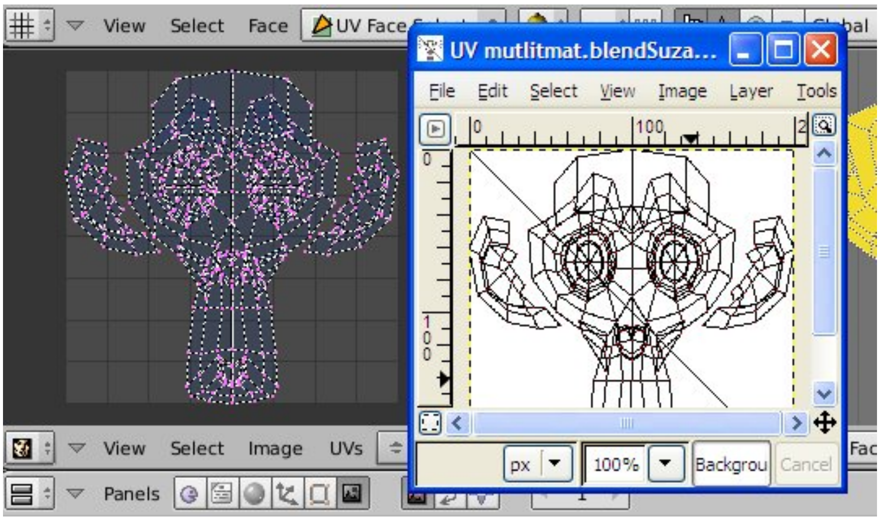
SVG
: if enabled, saves the file as a scalable vector graphic (SVG) format file, not a Targa.

Fill SVG faces
: fills in SVG faces to allow easier editing and face recognition

Edit
: Enabling Edit and specifying the name of an editing program int eh Editor: field will launch that program and load in the saved layout (after the image has been saved).

The image will be lines defining the UV edges that are within the image area of the UV mapping area. Edges outside the boundary, even if selected, will not be shown in the saved graphic.

The artist will use this as a transparent layer in their paint program as a guide when painting your texture. The example below shows Blender in the background, and the Gimp working on the texture, using the saved layout as a guide. Note that targa format supports the Alpha channel, so you can paint transparent areas of the mesh.



Using the layout as a guide in Gimp

Previous: [Manual/Unwrapping a Mesh] [Contents] *Next:* [Manual/Editing the UV Layout]

User Manual: Contents | Guidelines | Blender Version **2.43**

After unwrap, you need to arrange the UV maps into something that can be logically painted. Your goals for editing are:

- Arrange the different UV maps together into a coherent layout
- Stitch some pieces (UV maps) back together
- Minimize wasted space in the image
- Enlarge the 'faces' where you want more detail
- Re−size/enlarge the 'faces' that are stretched
- Shrink the 'faces' that are too grainy and have too much detail

With a minimum of dead space, the most pixels can be dedicated to giving the maximum detail and fineness to the UV Texture. A UV face can be as small as a pixel (the little dots that make up an image) or as large as an entire image. You probably want to make some major adjustments first, and then tweak the layout.

Major Adjustments

When you have unwrapped, possibly using seams, your UV layout may be quite disorganized and chaotic. You need to proceed in two steps: Orientation of the UV mapping, and then arranging the UV maps.

3D View: Face Mirror and Rotate UVs

Recall how the red and green outlines show the orientation of the UV Texture relative to the face? Well, you might find that, for example, the image is upside down or laying on its side. If so, use Face−>Rotate UVs (in the 3D window in Face Select mode) menu to rotate the UV layout in 90−degree turns. The Face−>Mirror UVs to flips the image over like a pankcake in a pan, mirroring the layout and showing you the image 'reversed'.

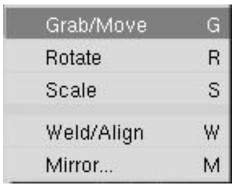
Merging UV Maps

Each time you unwrap, Blender creates a set of UV Coordinates shown in the UV/Image Editor window. So, if you unwrap 2 sides of a cube, and then 3 sides, and then the other remaining side of the cube, there will be 3 sets of UV Coordinates or UV Maps. You would unwrap this way if you wanted to use three different images for each of those maps. However, if you change your mind, simply select all the faces (for example, all 6), and re−use the Archimapper, the UV/Image Editor window's UVs−>Unwrap menu item. The maps will be recomputed to be non−overlapping.

Separating UV Maps

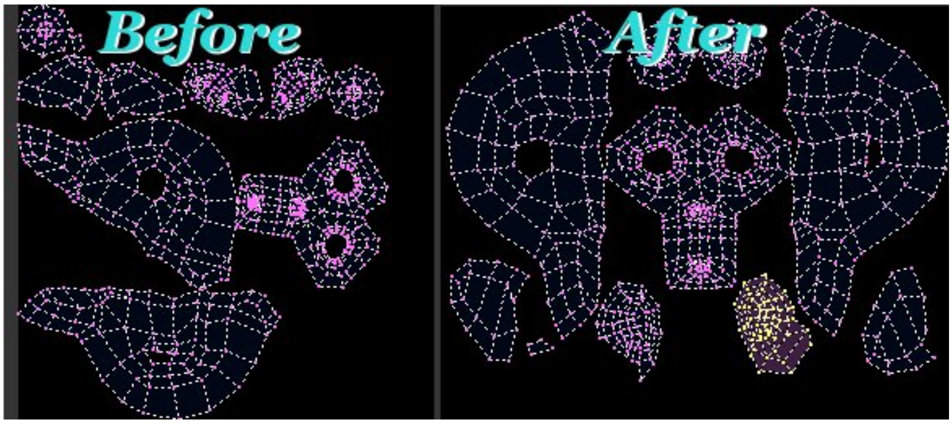
If you want to use a different image for a particular set of faces, but those faces are already mapped to a different UV Texture (image), simply select only those faces and unwrap them again. In the UV/Image Editor window a new map for them will be shown, and you can assign a new image to that map.

Arranging the UV Layout



UV Transformation Menu.

In the UV Editor you will see a representation of your selected faces as yellow or purple vertices connected with dotted lines. Wherever there was a seam or disconnected sub−mesh, there will be a UV map for that piece. During the unwrap, these pieces may be oriented illogically to you as a painter. You can use the same techniques here as in the Mesh EditMode to select, move, rotate, scale, and so on. With the Lock button pressed you will also see realtime feedback in 3D of what you are doing. Scaling and Translating of vertices can be done in the local X or Y axis of the map if needed. Just press X or Y after entering the scale command (S). Proportional editing is also available and it works the same way as in Edit Mode for meshes. Vertices in the UV Editor can be hidden or shown using the H and Alt H respectively, the same way as in Edit Mode.



Select Linked, Grab, and Rotate to Arrange layout

When arranging, keep in mind that the entire window is your workspace, but only the UV coordinates within the black square are mapped to the image. So, you can put pieces off to the side while you arrange them. Also, each UV unwrap is its own linked set of coordinates. You can lay them on top of one another, and they will onionskin (the bottom one will show through the top one). To grab only one though, RMB select one of the UV coordinates, and use Select−>Linked UVs (Ctrl L) to select connected UVs, not border select because UVs from both will be selected.

Deleting a UV Layout

A mesh can have multiple unwraps, each resulting in a UV map for that section of the mesh that was selected. You may get into a situation where you just want to delete everyting and start over. In the Buttons window, the Editing (F9) buttons, there is a Mesh panel. On that panel, click the fat X (Delete) button next to the name of the UV Texture you want to delete. If you delete every entry there, all the UV layouts for that mesh will be deleted and you will be kicked out of UV Face Select mode. The second that you re−enter UV Face Select mode, you create at least one base UV Texture.

UV/Image Editor Menu

When working with a UV Layout, 99% of your time will be spent in using the UV/Image Editor window. There are many options and features in this window to make you productive. The header consists of a menu,

an image selector and a few buttons. Some of the menu items and buttons are relevant when using UV textures. Only the ones relevant to changing the UV layout are discussed in this section.

View Menu

This menu controls how and what you see when working with UVs:

Maximize Window
: Very often, especially when painting, you will want to expand the window pane to full screen to see and work on the details. Or you can wear out your middle mouse button

View All
: Scales the image and UVs to fit within the window pane. Sometimes UVs can stray off into the woods, and this helps you find them and bring them back into the image area.

View Selected
: Centers the view on the selected faces

Update Automatically
: As you move UVs around, the resulting texture changes on the mesh (shown in the 3D window) are updated as you move around. Otherwise, updates are made when you drop the UVs. Use this option if you have the CPU power.

View Navigation
: Hotkeys to zoom in and out the display; same as using the MMB wheel. Also note the window can be panned using Shift MMB. There is no rotate or User view, since we are dealing with a 2 dimensional image.

Draw Shadow Mesh
: Toggles whether to draw an outline of the whole UV Layout for background painting alignment. Very Handy.

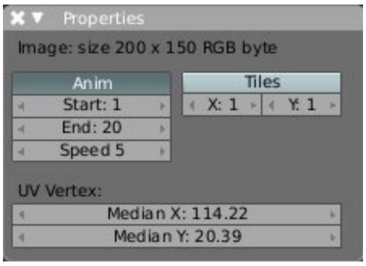
Draw Faces
: When enabled, draws selected faces over the image

Display Normalized Coordinates
: Displays the UV Coordinates in the Properties panel normalized to 1.0. For example, a UV coordiate with X=64 over a 256 grid would be normalized to 0.25.

Composite Preview
: *A work in progress...*

Curves Tool
: Shows the Combined, Red, Green and Blue (CRBG) Color Curves applet. Select a channel (C, R, B, or G) and adjust the curve to affect the colors of the image.

Paint Tool
: Shows the Vertex Paint floating window. See Vertex Paint.



UV Properties

Real−Time Properties
: Shows a floating window which allows you to set Animation and Tile info used in the Game Engine.
  ◊ **Anim** and **Tiles** − See Using UV Textures

◊ **UV Vertex** − Normally, the UV Vertex information is in pixels and the black workspace defaults to 256x256; with an image loaded, this display shows you the UV coordinates in relation to the exact image pixel location. With the menu item Display Normalized Coordinates turned on, the vertex location is scaled to 1.0. Origin is the bottom left hand corner of the black workspace. You can click into the X or Y field and manually set the location of a single UV or the median points of multiple UVs, or use the arrows to adjust their location in increments.

Properties

Many sources can be used for the image portion of the UV Texture. This panel shows you what you are using, and gives you basic controls to select, reload, turn TV interlacing fields on/off (Even or Odd), and do anti−aliasing on the image to smooth the image.

Select Menu

This menu helps you select UVs to work on:

Linked UVs ; Ctrl L

This menu item selects all UVs that are part of the same UV map. Recall that a map is made for every submesh and seamed part of the mesh, and is analogous to a piece of cloth. Selecting Linked UVs works similarly to the command in 3D View. It will select all UVs that are 'connected' to currently selected UVs.

Pinned UVs ; Shift P

You can pin UVs so they don't move between multiple unwrap operations. This menu item selects them all.

Unlink Selection ; Alt L

Cuts apart the selected UVs from the map. Only those UVs which belong to fully selected faces remain selected following this command. As the name implies, this is particularly useful to unlink faces and move them elsewhere. The hotkey is analogous to the mesh Separate command.

Select/Deselect All ; A

Selects or de−selects all UV coordinates. When initially unwrapping, you will want to select All UVs to rotate, scale, and move them around.

Border Select Pinned ; Shift B

Use the box lasso to select only pinned UV coordinates.

Border Select ; B

Use the box lasso to select normal UV coordinates.

Active Face Select

If turned on, when you RMB click a 'face' in the UV map, the corresponding face in the 3D View is highlighted. Use this as a nice way of seeing how the initial unwrap corresponds to your real world to aid in rearranging the map into a coherent layout. In addition, (stay with me here) all four UVs that correspond to that face are selected. Very handy for moving/rotating a whole UV face at a time, instead of individual UVs.

Image Menu

I know, I know, you're anxious to get started painting. Refer to the next section for help on this menu.

UVs Menu

This is a long menu that gets cut off sometimes. Here are the UV layout−related choices in the order that they appear:

Scripts

Displays the list of Python scripts you have loaded on your PC that do handy things with UV Textures:

- ◊ Auto Image Layout if you have one UV Texture for one part of the mesh, and another different UV Texture (map and image) for another part, use this script to merge them together.
- ◊ Save UV Face Layout to save a shadow outline of your UV layout; great as a guide layer when using external paint programs. Enable the SVG option to create a file that uses curves.
- ◊ Texture Baker and UV Painter are discussed in the next painting section.

Show/Hide Faces

Hide and show selected faces. Works the same way as for meshes in the 3D View, allowing you to focus only on certain parts of the layout.

Proportional Falloff and Proportional Editing

Same as mesh, and especially useful because often you are moving around a bunch of vertices in a tight space. Hotkey is O. Mousewheel to change the circle of influence, just like mesh editing.

Weld/Align

When you made a seam and unwrapped it, the mesh was cut at the seam into two pieces. Where they were cut apart, there are now two UV vertices for each mesh vertice; one for a corner or side of one UV map, and another for the other. In order to fill in the gaps between pieces, and make one continuous map, you can Weld them back together. Select the two vertices and then Weld them together. Align vertices in either the X or Y direction so that similar vertices line up. The Weld point is about halfway in between the two.

Mirror

Occasionally, a UV map will be backwards from your normal way of thinking. This menu option flips the map like a pancake, in either the X or the Y direction.

Transform

Grab/translate (G), scale (S) and rotate (R) selected vertices. Indespensible, and works just like meshes.

Stitch and Limit Stitch

Much like the mesh Remove Duplicates function, this function welds together corresponding vertices that are close. Not close as in horseshoes and hand grenades, but close as defined by the Limit. Different parts of a UV map can be stitched if the border UV vertices correspond to the same mesh vertices by using the Stitch command. The Stitch command works joining irregular outlines. Just select the vertices at the border line; if you are using the Stick UVs to Mesh Vertex, then the corresponding coordinates on the other UV Map will also be selected. Stitch (V) snaps together UVs, whatever the distance between them, whereas Limit Stitch (Shift V)only welds UVs within a given range (Limit:, **20** pixels by default). Its advantage over Weld is that it prevents UVs, that are supposed to stay separate, from being joined together.

Minimize Stretch

Very often, the initial unwrap will have some vertices bunched up right next to some that are spread apart. Much like smoothing out a wrinkled paper, this handy function spreads out bunched up selected vertices. A large face, at an angle to the projection, will result in a small layout. Hence, any image will appear stretched out, like a painting on a balloon. Use this option to relax the stretch a little.

Pin and Unpin

Using the Pin (P) command on selected vertices forces them to stay put between multiple unwrap operations, pinned in their current location. Any subsequent Unwraps will not move them. They appear red and larger than other UV coordinates, like a pushpin. Unpin (Alt P) sets them free. Read more about it in the [#Using the Pin command] section.

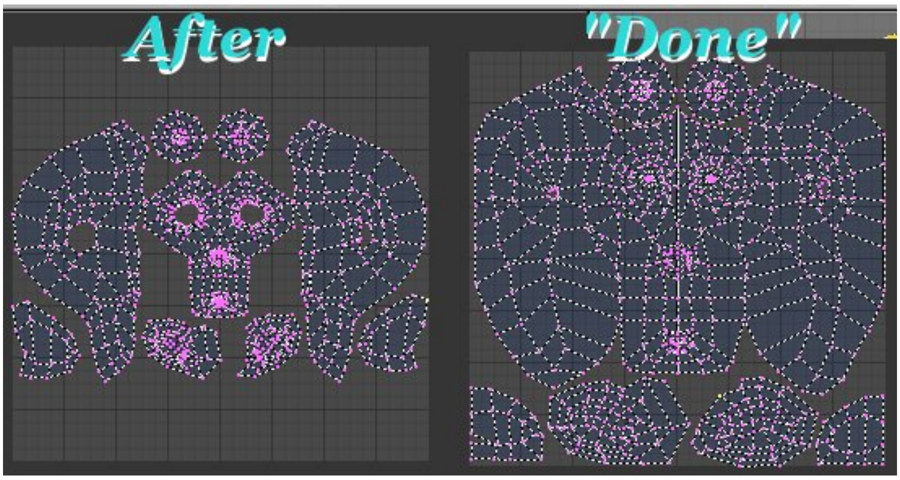
Layout Clipped to Image Size
: Keeps the UVs in the corral of the Image size. Prevents you from moving a UV outside the image area.

Quads Constrained Regular
: Too geeky for me. Sorry. I mean really, those three words just don't go together in any comprehensible way for my tiny brain :) (In fact, I think this means "Quadrangles (are) constrained (to be) regular"â€¦) When enabled and you grab a UV coordinate, it tries to help you make the UV area square or rectangular. Use it when texturing an orthogonal image to a real−world mesh. Warning: it does an automatic unlink from neighbors if they aren't in line as well.

Snap to Pixels
: Put in here by the developers for the perfectionist in all of us; it causes dropped UVs to snap to an even pixel location, so that the UVs line up perfectly with the image. Exactly. Precisely. NO .126 or .436 or any garbage like that. Nice clean units. Gotta love it.

Iteration and Refinement

At least for common people, we just don't "get it right the first time." It takes building on an idea and iterating our creative process until we reach that magical milestone called "Done." In software development, this is called the Spiral Methodology. Applied to Computer Graphics, we cycle between modeling, texturing, animating, and then back to making some modifications to mesh, re−UV mapping, tweaking the animation, adding a bone or two, finding out we need a few more faces, so back to modeling, etc. We continue going round and round like this until we either run out of time, money, or patience, or, in some rare cases, are actually happy with our results.



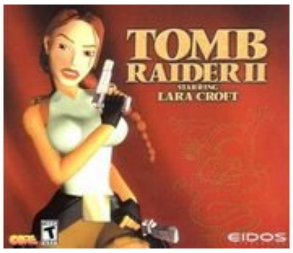
Maximum space for detail areas using Stitch, Align, Minimize Stretch, Scale selected

Now would be a really good time to save an outline of your UV layout. Don't be afraid to try different UV Calculations for the same set of faces to get one that looks good to you and will not be a lot of work to arrange.

Tradeoffs

For example, consider the creation of a game character. We want the character to look realistic. Realism in appearance for game characters comes in two parts: shape and texture. Regarding shape, we want the model to look proportioned and smooth, and not too blocky. But the more we smooth it out and add features, the more polygons and faces we have. More polys is bad, because it slows down game play because it taxes computer resources. Texture adds realistic colors to the surface, but again, more detailed images means more pixels and

thus taxes compute power again. So, there is a balance between the number of faces and the size of the textures, and acceptable realism and game play. Very early games, for example, had a human head the shape of a blocky cube with a texture of a face; let's just say you had to use your imagination. But, it achieved an acceptable level of realism *and* had responsive game play on computers available at the time.



Later on, early 'Tomb Raider' PC games by Eidos are (in my humble opinion) an excellent example of portraying a shapely character in as few polys as possible with small but creative images that faked shadows and creases. Game play was fast, responsive, and a few sharp corners could be overlooked.

Using the Pin command

Developing a game character is thus an iterative creative process. We model the game character, unwrap it, start creating an image for it, and then realize that we have too many faces. We reduce the faces, unwrap again, and start repainting.

This is where the Pin command comes in. The first time we unwrap and start drawing an image, (for example the clothes), we want to preserve that work. When we unwrap the second time, we don't want the chest UVs to be mapped somewhere else on the image; we want them to stay there, right over the painting of the vest. So, before unwrapping the second time, we Pin them, then change the mesh, and unwrap the mesh again. The second time we unwrap, those pinned UVs coordinates will stay in place, no matter what UV Calculation method we use or how we change seams.

The other use of the Pin command is sort of an undo. Imagine that when editing the UV map, you discover that yesterday you accidentally welded five unrelated UV coordinates together by mistake. Undo is therefore not available, and you don't want to have to go back to an old copy. Don't feel stupid; your author did so in generating these examples. So you pin all the coordinates except the welded one, re−unwrap, and the UV coordinates will be 'restored'.

Refining the Layout

Refinement comes into play when we finally look at our character, and realize that we need more detail in a particular spot. For example, areas around the eyes might need crow's feet, or we need to add a logo to the vest. As you start to edit the image, you realize that there just aren't enough pixels available to paint the detail that you want. Your only choice is to expand the size (scale out) that UV face. Using the minimize stretch or scale commands, you expand the UV faces around the eyes or chest, allocating more pixels to those areas, but at the same time taking away pixels (detail) from something else, like the back of the head. After refining the UV map, you then edit the image so that it looks right and contains the details you want.

Reusing Textures

Another consideration is Re−Use. Each image file is loaded in memory. If you can re−use the same image on different meshes, it saves memory. So, for example, you might want to have a generic 'face' painting, and use that on different characters, but alter the UV map and shape and props (sunglasses) to differentiate. You might want to have a 'faded blue jeans' texture, and unwrap just the legs of characters to use that image. It would be good to have a generic skin image, and use that for character's hands, feet, arms, legs, and neck. When modeling a fantasy sword, a small image for a piece of the sword blade would suffice, and you would Reset Unwrap the sword faces to re−use that image down the length of the blade.

Previous: Manual/Working with UV	Contents	*Next:* Manual/Using UV Textures

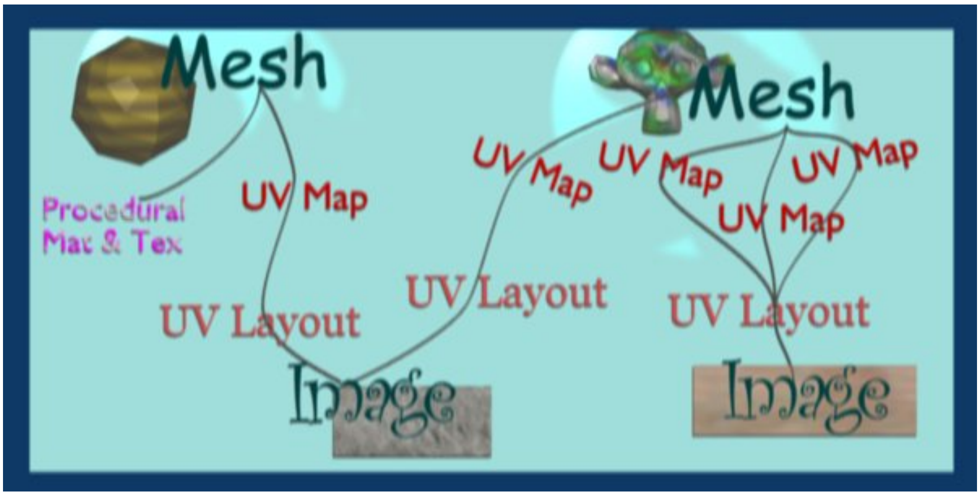
User Manual: Contents | Guidelines | Blender Version **2.44**

Up to this point, we have only talked about half of UV Textures; the UV map and layout. The layout is the arrangement of all the UV maps. Each UV map 'maps' image pixels to a mesh face. There is one UV map for each seam or sub−mesh. The entire layout is colored by an image. Blender provides several features that help you when working with the image part of the UV Texture. Blender also includes a built−in texture painting program. This section discusses how to use images effectively.

Goals

Texture images take up precious memory space, often being loaded into a special video memory bank that is very fast and very expensive, so it is often very small. So, keep the images as small as possible. A 64x64 image takes up only one fourth the memory of a 128x128 image. For photo−realistic rendering of objects in animations, often larger image textures are used, because the object might be zoomed in on in camera moves. In general, you want to use a texture sized proportionally to the number of pixels that it will occupy in the final render. Ultimately, you only have a certain amount of physical RAM to hold an image texture and the model and provide work space when rendering your image.

If you can re−use images across different meshes, this greatly reduces memory requirements. You can re−use images if you map those areas of the meshes that 'look alike' to a layout that uses the common image. In the overview below, the left image is re−used for both the sphere and a portion of the monkey. The monkey uses two layouts, one which has one UV map of a few faces, and another that has three maps.



How all the parts of UV Texturing work together

You also do not have to UV map the whole mesh; the sphere above on the right has some faces mapped, but other faces use procedural materials and textures. Only use UV Textures for those portions of your mesh where you want very graphic, precise detail. For example, a model of a vase only needs UV Texture for the rim where decorative artwork is incorporated. A throw pillow does not need a different image for the back as the front; in fact many throw pillows have a fabric (procedural material) back.

As another example, you should UV map both eyes of a head to the same image (unless you want one bloodshot and the other clear). Mapping both sides of a face to the same image might not be advisable, because the location of freckles and skin defects are not symmetrical. You could of course change the UV map for one side of the face to slightly offset, but it might be noticeable. Ears are another example where images or section of an image can be mapped to similar faces.

Workflow

The process is to:

1. Create the Mesh. Unwrap it into one or more UV Layouts
2. Create one or more Materials for the Mesh
3. Create one or more images for each UV Layout and aspect of the texture
    1. Paint directly on the mesh using Texture Paint in the 3D window
    2. Load and/or edit an image in the UV Editor window
    3. Bake the existing materials into an image for the UV Editor window
4. Apply those images as UV Textures to the mesh to affect one or more aspects of the mesh
    1. Map to Color to affect the diffuse coloring of the mesh
    2. Map to Nor to give the surface a bumpy or creased look
    3. Map to Spec to make certain areas look shiny and oily
    4. Many other Map To options are available for photo−realistic results
5. Layer the UV Textures to create a convincing result.

Using Images and Materials

For an image to use as the color and alpha (transparency) of the texture, you can create an image in an external paint program and tell the UV/Image Editor to Open that file as the texture, or create a New image and save it as the texture.

If you want to start off by creating an image using an external paint program, you will want to save an outline of your UV faces by using the script Save UV Face Layout located in the UVs menu. This script was discussed here.

Creating an Image Texture

To create an image within Blender, you have to first create a New Blank Image with a uniform color or test grid. After that, you can color the image using the:

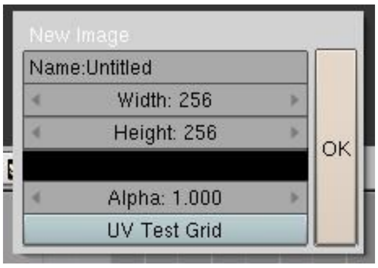
- Vertex colors as the basis for an image
- Render Bake image based on how the mesh looks in the scene
- Texture Baker script (discontinued past version 2.43)

After you have created your image, you can modify it using Blender's built−in Texture Paint or any external image painting program.

See Texture in 3D View but does not Render

You may be able to see the texture in Textured display mode in the 3D View; this is all that is required to have textures show up in Blender's Game Engine. Rendering, however, requires a material. You must have a TexFace material assigned to the mesh for it to render using the UV Texture. In the Material settings, ADD NEW material to a selected object and enable TexFace.

Creating a New Blank Image



Once you have unwrapped, use the UV/Image Editor Image−>New to create a new image that will be the UV Texture.

Name
: Give your image a more descriptive name

Width and Height
: If you are texturing an item for a game, it is best to use image sizes that are powers of two (16, 32, 64, 128 ...) for both width and height, so they can be drawn properly in real−time using OpenGL. Most 3D cards don't support images larger than 2048x2048 pixels.
: For rendered artwork, textures can be any size. Images do not have to be square; they can be any size that you want, provided you have the memory and graphics display capability. Size should be based on the amount of detail that you need.

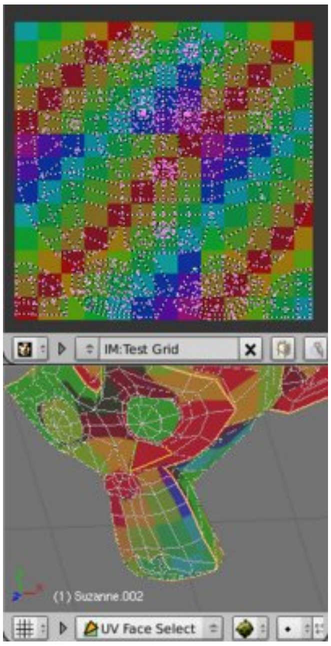
Base Color
: Click on the color swatch (default is black) to bring up the color picker applet, and choose the starting base color.

Alpha
: Choosing an alpha less than 1 will allow other layers of UV textures to show through. The key to realistic looking skin, for example, is many layers that can be individually controlled.

UV Test Grid
: Click this to use the UV Test grid discussed below.

Of course, this simplest of new images will be blank. You will have to start texture painting to get some color into those cheeks. ESC aborts the new image creation.

Using the Test Grid



Use the UV Test Grid option to check for undue stretching or distortion of faces. If your image is a base uniform pattern and you want the application of that image to your model to look like cloth, you do NOT want any stretching (unless you want the cloth to look like spandex).

When you render, the mesh will have the test grid as its colors, and the UV Texture will be the size image you specified. You can save the UV image using the Image−>Save menu.

Modifying your Image Texture

To modify your new Texture, you can:

- UV Painter script to create an image from vertex colors
- Render Bake an image based on how the mesh looks
- Texture Baker script (version 2.43−) to create an image
- Paint using Texture Paint
- Use an outside paint package to create an image.

The first three options, (UV Painter, Render Bake, and Texture Baker) replace the image with an image that they create. Texture paint and outside packages merely add or enhance the image. Regardless of which method you use, ultimately you must:

- Save your texture in a separate image file (for example JPG for colors, PNG with RGBA for alpha)
- Pack the image inside the blend file (UV/Image Editor Image−>Pack as PNG)

The advantage to saving as a separate file is that you can easily switch textures just by copying other image files over it, and you can use external editing programs to work on it. The advantage to packing is that your whole project is kept in the .blend file, and that you only have to manage one file.

Creating an image from Vertex Colors via UV Painter Script

Removed in Blender ver 2.43, TODO: Equivalent feature?

The UV Painter script is located in the UV/Image Editor UVs menu. This nifty script takes the vertex colors from the selected faces and colors in your UV layout. The line button draws an outline of your UV map, and the Scale button scales up or down the image. When you Save your image, a targa−format image file is created for your editing pleasure. UV Painter is an easy way to start painting your mesh, once you have finished your layout.

The Redraw button updates the painting based on any UV layout changes or vertex painting done since the last time the cursor visited the window, although updates may occur automatically.

A small (bug?) Note: After saving the targa image, you must edit it with an external program (Gimp) as it saves dead space off to the side, and you want to scale it in pixels.

Replacing an image via Render Bake

The Render Bake feature provides several tools to replace the current image based on a render of:

- Vertex Paint colors
- Normals (bumps)

- Procedural materials, textures and lighting
- Ambient Occlusion

Click on the hyperlink above for more info. The Render Bake produces an image, already mapped to your UV layout, shown in the UV/Image Editor.

Creating an image via Texture Baker Script

Removed in Blender ver 2.43, TODO: Equivalent feature?

V.: 2.43

The Texture Baker script, available from the UV/Image Editor UVs menu, saves a UV texture layout of the chosen mesh, that can be used as a UV map for it. It is a way to export procedural textures from Blender as normal image textures that can be edited with a 2d graphical image manipulation program or used with the mesh in games and other 3d applications.

Select your faces in the 3D View Face Select mode; run the script by selecting the menu item. A pop−up panel will appear, allowing you to change the image name or leave it be. The first time you run the script, supply the filename. Choose a reasonable resolution, and an image is rendered. The image depends on your Material settings:

- If you do not have either VCol Paint or TexFace enabled in your material buttons, you will get the UV Layout colored with the procedural (base) material and textures that are current; for example a shadeless purple that is marbled.
- With VCol Paint enabled, the rendered image includes the procedural materials and textures, modulated by the vertex paint job.
- Selecting both VCol Paint and TexFace incorporates procedurals, vertex, and current UV image, melding them all into one beautiful render.

Save the image using Blender's File−>Save Image dialog, and the image will be saved in the format specified in the render settings. You can then load it as discussed below.

Broken in 2.44

In 2.43 and before, if the script does not run completely perfectly with you answering all questions, it may leave all Layers unselected, with your 3D View thus blank. Don't know why, it just does.

Create/Modify via an External Image Paint Program

Using your favorite image painting program, draw something that matches your UV layout. Then save your changes, and back in Blender, use the Image−>Open menu command to load it as your UV image for the mesh in Face Select Mode for the desired (and active) UV Texture layer.

Modify an image via Texture Paint

Use the UV/Image Editor menu *Image−>New*. Then start painting your mesh with Texture Paint.

Using a Saved Image

Use Image−>Open to search for and use images in any popular format as the UV Texture. It will be opened and placed as a background to the layout. More often than not, use an image that follows your UV layout. Recall that you saved an outline of your layout and used that to guide the painting of your UV texture image.

If you open an avi file, the first frame of the animation will be used. You can not open an image sequence or use a frame in the middle of a file.

Finding Images

When you open the .blend file, Blender goes out and loads in the most recent image from its location on your hard drive.

In a team environment, or if you are using an external paint program to edit the image while the .blend file is active, and the file is updated and re−saved, use the UV/Image Editor to Image−>Reload it and see the latest and greatest in Blender. Also, use Reload if you have mapped more faces to an image, and the 3D View will be updated with the latest image mapping back to faces.

If you move the image file, Blender may not be able to find it, and you will have to Image−>Replace it. Use this option to map a UV layout to a different image altogether.

Organize your images

In any scale project, there quickly becomes hundreds of images that are used as UV Textures. Set up a //tex/UV/ directory to hold your UV Textures. Practice good directory and change management. Unfortunately, Blender does not support the Windows shortcut links.

Blender contains a Find Image Target Paths script in the UVs menu in the UV/Image Editor window. Starting from a specified *root* directory for your project, this script will search down and, based on filename, reload images automatically. Use this script if you have renamed a subdirectory or moved a few images around inside your project.

The other way to locate images is to change one of your Blender UI panes to an Image Browser window. This file browser showns you thumbnails and image information (size, format, etc.) of only image files in a directory.

Mapping the Image Texture

Some Modifiers Prevent UV Mapping

In particular, the Decimate Modifier, even if it is only in the Editing modifier list (stack) and not actually applied to the mesh, prevent UV mapping, since it affects the number of vertices and thus UV coordinates.
You then have to create a new Material for the mesh. Then you have two ways of applying that texture to the material:

- The proper way is to map the image using the UV texture, and to load that image as an image texture.
- The quick way is to enable TexFace in the Material panel. This tells Blender to use the UVTexture as the base material color (alpha is ignored). Any other textures are layered on top of that base from the top texture channel to the bottom according to their mix method. For example, a wood texture mapped to Alpha on top of a TexFace image makes streaks of the material transparent.

To map the image as a Texture channel (usually the top channel), in the Map Input sub−panel enable UV, and enter the name of the UV Texture (by default UVTex). The advantage is many more options in the Texture's Map Image panel, such as UseAlpha and repeat/mirror. You can also control how multiple textures are layered onto one another by their order in the Texture channels, as well as how they mix with each other, animate their color influence, etc.

The previous pages explained how to create a set of UV Layouts for different portions of the mesh. For example, there may be one UV Layout for the face of a character, and another for their clothes. Now, to texture the clothes, you need to create an image at least for the Color of the clothes, and possible a "bump" texture to give the fabric the appearance of some weave by creating a different image for the Normal of the clothes. Where the fabric is worn, for example at the elbows and knees, the sheen, or Specularity, of the fabric will vary and you will want a different image that tells Blender how to vary the Specularity. Where the fabric is folded over or creased, you want another image that maps Displacement to the mesh to physically deform the mesh. Each of these are examples of applying an image as a texture to the mesh.

As another example, the face is the subject of many questions and tutorials. In general, you will want to create a Material that has the basic skin color, appropriate shaders, and sub−surface scattering. Then you will want to layer on additional UV Textures for:

- Freckle map for Color and Normal aspects
- Subdermal veins and tendons for Displacement
- Creases and Wrinkles and skin cell stratification for Normal
- Makeup images for Color
- Oily maps for Specularity
- For a zombie, Alpha transparency where the flesh has rotted away
- Under chin and inside nostrils that receive less Ambient light
- Thin skin is more translucent, so a map is needed for that

Each image is mapped by using another Texture Channel. Each of these maps are images which are applied to the different aspects (Color, Normal, Specularity) of the image. Tileable images can be repeated to give a smaller, denser pattern by using the Texture controls for repeat or size.

Replacing the active Image

Recall that each face gets coordinates and a link to an image. To map a face to a different image, simply select that face (or faces) and use the UV/Image Editor window Image menu to Replace the current image with an existing file (such as a JPG or PNG file).

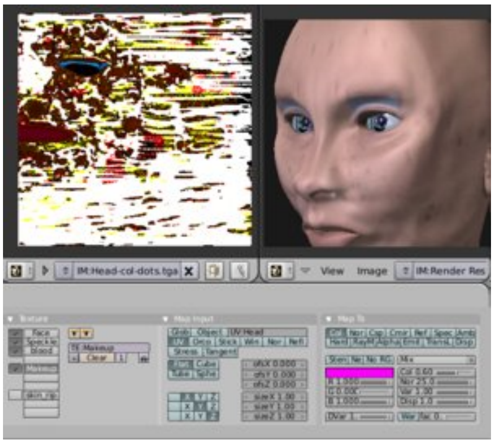
Packing Images inside the Blend file

If you pack your .blend file, the current version of all UV Texture images are packed into the file. If those files later change, the updates will not be automatically re−packed; the old version of the image is what will be used. To update, you will have to re−pack or reload.

The File−>Append function automatically goes into .blend files and shows you the image textures packed in it. The public domain Blender Texture CD is also a great resource, and there are many other sources of public domain (and licensed) textures. All textures on the Elephant's Dream CD are public domain. And if it looks like a duck and quacks like a duck...

Layering UV Textures

Base UV Texture



Layered UV Texture

Great textures are formed by layering images on top of one another. You start with a base layer, which is the base paint. Each successive layer on top of that is somewhat transparent to let the bottom layers show through, but opaque where you want to add on to details.

To avoid massive confusion, all image textures for a mesh usually use the same UV map. If you do, each image will line up with the one below it, and they will layer on top of one another like the examples shown to the right. To do this, just create one UV Texture (map) as described in this section. Then, create material image textures as described in the procedural materials section. Instead of mapping to Original Coordinates (OrCo), map to UV. Use that map name repeatedly in the Material−>Textures−>Map Input panel by selecting UV and typing the name in the text field. In the example to the right, our UV Texture is called "Head" (you may have to expand the image to see the panel settings). Then, the image texture shown will be mapped using the UV coordinates. In the "Base UV Texture" example to the right, the face has two textures UV mapped; one for a base color, and another for spots, blemishes and makeup. Both textures use the same UV Texture map as their Map Input, and both affect Color. The Makeup texture is transparent except where there is color, so that the base color texture shows through. Note that the colors were too strong on the image, so they amount of Col affects is turned down to 60% in the second layer (the blemish layer).

Normally, we think of image textures affecting the color of a mesh. Realism and photo−realistic rendering is a combination of many different ways that light interacts with the surface of the mesh. The image texture can be Mapped To not only color, but also Normal (bumpiness) or Reflection or any of the other attributes specified in the Map To panel. If you paint a grey−scale image (laid out according to the UV Layout) with white where the skin is oily and shiny, and dark where it is not, you would map that input image according to the UV Layout, but have it affect Specularity (not color). To make portions of a mesh transparent and thus reveal another mesh surface underneath, you would paint a grey−scale image with black where you want the texture transparent, map input to UV, and map it to Alpha (not color). To make portions of a mesh, like a piece of hot metal, appear to glow, you would use a grey−scale image mapped to Emit.

Mix and Match Materials

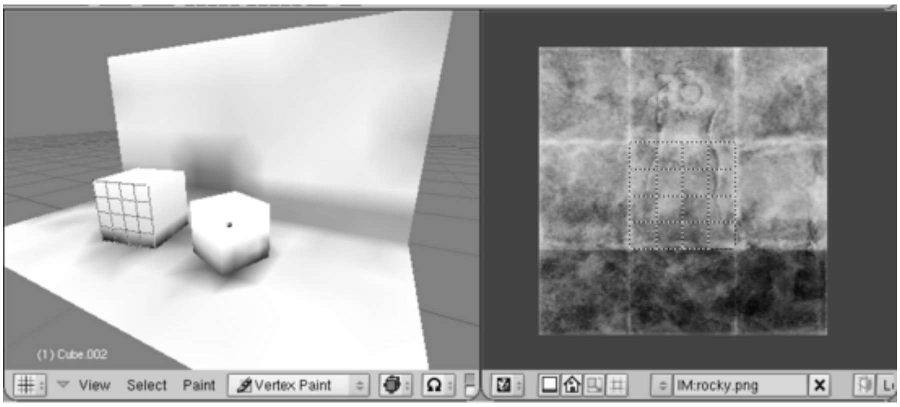
You can mix and match procedural materials and textures, vertex paint, and UV textures onto the same mesh.

The image to the right has a world with a red ambient light. The material has both VCol Paint and TexFace enabled, and receives half of ambient light. A weak cloud texture affects color, mixing in a tan color. The right vertices are vertex painted yellow and the left is unpainted procedural gray. The UV Texture is a stock arrow image from the public domain texture CD. Scene lighting is a white light off to the right. From this information and the User Manual thus far, you should now be able to recreate this image. In other words, I have taught you all that I know, my young padawan. May the Force be with you. Oh wait, there's more...

You can also assign multiple materials to the mesh based on which faces you want to be procedural and which you want to be texture−mapped. Just don't UV map the faces you want to be procedural.

You can use UV Textures and VertexPaint (V in the 3D View window) simultaneously, if both are enabled in the Material settings. The vertex colors are used to modulate the brightness or color of the UV image texture:

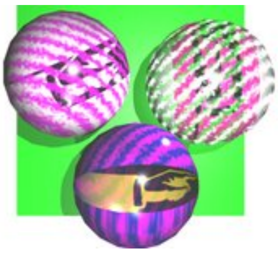
- UV Texture is at the base
- Vertex paint affects its colors, then
- Procedural textures are laid on top of that,
- Area lights shine on the surface, casting shadows and what not, and finally
- Ambient light lights it up.



Vertex colors modulate texture.

A UV Layout can only have one image, although you can tile and animate the image. Since a layout is a bunch of arranged UV Maps, and a UV Map maps many mesh faces, a face can therefore only have one UV Texture image, and the UV coordinates for that face must fit entirely on the image. If you want a face to have multiple images, split the face into parts, and assign each part its own image.

Using Alpha Transparency



Alpha UV Textures

Alpha 0.0 (transparent) areas of a UV Image render as black. Unlike a procedural texture, they do not make the base material transparent, since UV Textures do not operate on the base procedural material. The UV texture overrides any procedural color underneath. Procedural Textures are applied on top of UV Textures, so a procedural image texture would override any UV Texture. Transparent (black) areas of a procedural texture mapped to alpha operate on top of anything else, making the object transparent in those places. The only thing that modulates visible parts of a UV Texture are the Vertex Colors. In the example to the right, the finger image is transparent at the cuff and top of the finger and is used as a UV Texture. All three balls have a base material of blue and a marbling texture. The base material color is not used whenever TexFace is enabled.

The top left ball has not had any vertex painting, and the finger is mapped to the middle band, and the texture is mapped to a pink color. As you can see, the base material has VCol Paint and TexFace enabled; the base color blue is not used, but the texture is. With no vertex painting, there is nothing to modulate the UV Texture colors, so the finger shows as white. Transparent areas of the UV Image show as black.

The top right ball has had a pink vertex color applied to the vertical band of faces (in the 3D View window, select the faces in UV Paint mode, switch to Vertex Paint mode, pick a pink color, and Paint−>Set Vertex Colors). The finger is mapped to the middle vertical band of faces, and VCol and TexFace are enabled. The texture is mapped to Alpha black and multiplies the base material alpha value which is 1.0. Thus, white areas of the texture are 1.0, and 1.0 times 1.0 is 1.0 (last time I checked, at least), so that area is opaque and shows.

Black areas of the procedural texture, 0.0, multiply the base material to be transparent. As you can see, the unmapped faces (left and right sides of the ball) show the vertex paint (none, which is gray) and the painted ones show pink, and the middle stripe that is both painted and mapped change the white UV Texture areas to pink. Where the procedural texture says to make the object transparent, the green background shows through. Transparent areas of the UV Texture insist on rendering black.

The bottom ball uses multiple materials. Most of the ball (all faces except the middle band) is a base material that does not have TexFace (nor Vertex Color Paint − VCol Paint) enabled. Without it enabled, the base blue material color shows and the pink color texture is mixed on top. The middle band is assigned a new material (2 Mat 2) that *does* have vertex paint and TexFace enabled. The middle band of faces were vertex painted yellow, so the white parts of the finger are yellow. Where the pink texture runs over the UV texture, the mixed color changes to green, since pink and yellow make a green.

If you want the two images to show through one another, and mix together, you need to use Alpha. The base material can have an image texture with an Alpha setting, allowing the underlying UV Texture to show through.

To overlay multiple UV images, you have several options:

- Create multiple UV Textures which map the same, and then use different images (with Alpha) and blender will overlay them automatically.
- Use the Composite Nodes to combine the two images via the AlphaOver node, creating and saving the composite image. Open that composited image as the UV Texture.
- Use an external paint program to alpha overlay the images and save the file, and load it as the face's UV Texture
- Define two objects, one just inside the other. The inner object would have the base image, and the outer image the overlaid image with a material alpha less than one (1.0).
- Use the Material nodes to combine the two images via the AlphaOver or Mix node, thus creating a third noded material that you use as the material for the face. Using this approach, you will not have to UV map; simply assign the material to the face using the Multiple Materials

UV Textures vs. Procedural Textures

A Material Texture, that has a Map Input of UV, and is an image texture that is mapped to Color, is the same as a UV Texture. It provides much more flexibility, because it can be sized and offset, and the degree to which it affects the color of your object can be controlled in the Map To panel. In addition, you can have different images for each texture channel; one for color, one for alpha, one for normals, one for specularity, one for reflectivity, etc. Procedural textures, like Clouds, are INCREDIBLY simple and useful for adding realism and details to an image.

UV Texture	Procedural Texture
Image maps to precise coordinates on the selected faces of the mesh	Pattern is generated dynamically, and is mapped to the entire mesh (or portion covered by that material)
The Image maps once to a range of mesh faces specifically selected	Maps once to all the faces to which that material is assigned; either the whole mesh or a portion
Image is mapped once to faces.	Size XYZ in the MapInput allows tiling the texture many times across faces. Number of times depends on size of mesh
Affect the color and the alpha of the object.	Can also affect normals (bumpiness), reflectivity, emit, displacement, and a dozen other aspects of the mesh's appearance; can even warp or

	stencil subsequent textures.
Can have many for a mesh	Can be layered, up to 10 textures can be applied, layering on one another. Many mix methods for mixing multiple channels together.
Any Image type (still, video, rendered). Preset test grid available	Many different presents: clouds, wood grain, marble, noise, and even magic.
Provides the UV layout for animated textures	Noise is the only animated procedural texture
Takes very limited graphics memory	Uses no or little memory; instead uses CPU compute power

So, in a sense, a single UV texture for a mesh is simpler but more limited than using multiple textures (mapped to UV coordinates), because they do one specific thing very well: adding image details to a range of faces of a mesh. They work together if the procedural texture maps to the UV coordinates specified in your layout. As discussed earlier, you can map multiple UV textures to different images using the UV Coordinate mapping system in the Map Input panel.

UV/Image Editor Menu

There are several menu items, options and features in the UV/Image Editor window that pertain to using and manipulating the images used in UV Textures. In the View−>Properties panel, the Anim and Tile options are for the Blender Game Engine only. Animated UV textures are procedural textures that are mapped to UV coordinates.

Window Header Buttons

Pack
: The button that looks like a little package automatically puts a copy of the image file inside your .blend file when you use Image−>Open, automatically packing them all together. Use this option to transport only one file (instead of the .blend and all the .png's used), or to isolate your .blend from any changes that may be happening.

Rotation/Scaling Pivot

- **2D Cursor**; , : As analogous to the Object Mode/ Edit Mode 3D Cursor pivot. This will rotate the current UV selection around the 2D cursor position. LMB click to position the 2D cursor at desired location.
- **Median Point**; Shift , : As analogous to the Edit Mode Median Point pivot. Enabling the Median Point pivot will calculate a rotation point for a UV selection accordingly to the vertice count or weight of objects. Blender supposes every vertex has the same weight.
- **Bounding Box Center**; . : As analogous to the Edit Mode Bounding Box Center pivot. Blender will encompass your UV selection as tightly as possible with a 2D rectangle or box, and then use it's center as a rotation point.

Sync UV and Mesh Selection
: This will show all UV faces of your Unwrap/UV Map in the UV/Image Window, not only the selected faces. It is very useful when working (applying transformations) on your UV Map layout, or simply for a better perspective of how faces are linked and laying flat in the UV Map.

UV Vertex select mode
: Enables selection of vertice.

UV Face select mode
Enables selection of faces.
Sticky UV selection

- **Shared Vertex** ; Ctrl C : When welding or stitching, you want to be sure that you sew the UV maps together correctly. At a seam, when unwrapping, two UV coordinates were created, one for each side of the seam. With Stick UVs enabled, a RMB click will not only select the UV vertex closest to the mouse cursor, but also all the other UV vertices that correspond to the same mesh vertex (but who are now shown on the 'other' side of the seam in another UV map).
- **Shared Location** ; Alt C : Works in the same way, but only on the UVs that are 'connected', meaning they are within a 5 pixel range of the first selected UV.
- **Disable** ; Shift C : Disables sticky selection.

Snap while CTRL is held during transformation ; Shift TAB
When applying a transformation to a UV selection, hold down CTRL to snap the selection to your mouse pointer, according to the selected snap method chosen (described below). Once satisfied with position, LMB click to apply transformation or RMB click to cancel.

- **Median** : Move the median of the selection to the snap target (mouse pointer).
- **Center** : Move the current transformation center to the snap target. Can be used in conjunction with 2D Cursor to snap with offset.
- **Closest** : Move the closest point of the selection to the snap target.

Texture Painting
The magic pencil button changes your mouse and keyboard into a mini−paint program. Paint using the LMB . See Manual/Texture Paint
Draw With Alpha
UV Textures do not have to be totally opaque; they can be *partially transparent, like a partial reflection on a window. Turning on this button makes the display show only the opaque parts of the image.
Draw Alpha Only
The dot button disregards colors and shows the alpha channel of the image as a BW gradient, with white being Alpha of 1.0 which is totally opaque and black, which is totally transparent. Use this option to see what parts of the object will be transparent based on the UV Texture.
Lock
When changes happen in this window (UV/Image Editor), other affected windows are updated in real time.

Full Screen

When working on details, remember that you can expand any window to full−screen by pressing Shift Space to toggle the active window between a pane and full−screen. You can also use Ctrl and Ctrl

Image Menu

This menu gives you options when working with the image that is mapped to the mesh.

- **Realtime Texture Mapping**: sets display updates to either UV Coordinates (default) or Reflection. With reflection, the texture shown is like looking in a mirror at the image. Use this when rendering a scene featuring a mirror that has a reflection 'out the window' when there is no 'outside'. This is sometimes also called using the UV Layout as a *Reflection Map*, namely mapping the image as if it was a reflection. The image you want to use is what would be reflected, namely the view from the reverse camera angle.
- **Texture Painting**: Enables and turns on Manual/Texture Paint.

- **Pack Image**: When selected, takes all the images in use and puts a copy inside the .blend file. Use this when sending off your file or packing up for the weekend.
- **Reload**: refreshes the image in Blender by re−reading the source image file. Use this if the artist has updated the image with changes.
- **Replace**: Replaces the Image with a new one that has a different name/location, keeping the UV mapping. The old one is discarded from memory/pack. Use this if you mistakenly opened the wrong file.
- **Save As**, **Open**, and **New**: saves the current image, opens an existing file, and creates a new image, respectively. But I bet you already figured that out.

Thumbnails

Blender has a built−in picture window, the Image Browser window type, that allows you to scroll through directories on your hard disk, and shows you thumbnails only of image files in the directory. When you hover over a file, the header shows you the size and format of the file. Very Handy...even better than the Windoze file browser.

UVs Menu (Image Related)

Sometimes it is necessary to move image files to a new location on your hard disk. Use the Find Image Target Paths script to update the image links. You can fill in the top level root directory name, and Blender will search down from there to find a similar file.

Previous: Manual/Editing the UV Layout	Contents	*Next:* Manual/Texture Paint

User Manual: Contents | Guidelines | Blender Version **2.43**

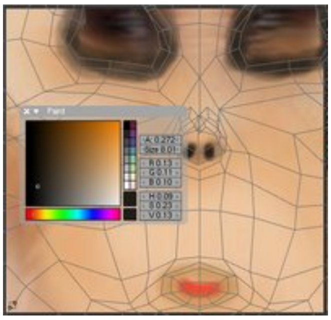
A UV Texture is an image that is a picture (image, sequence or movie) that is used to color the surface of a mesh. The UV Texture is mapped to the mesh through one or more UV maps. There are three ways to establish the image used by the UV Texture.

- Paint a flat image in the UV/Image Editor onto the currently select UV Texture, using its UV map to transfer the colors to the faces of the mesh
- Paint the mesh in the 3D View, and let Blender use the currently selected UV map to update the UV Texture
- Use any image−editing (paint) program to create an image. In the UV/Image Editor, select the UV Texture and load the image. Blender will then use that texture's UV map to to transfer the colors to the faces of the mesh

Blender features a built−in paint mode, called Texture Paint, designed specifically to help you edit your UV Textures and Images quickly and easily in either the UV/Image Editor window or the 3D View window. Since a UV Texture is just a special−purpose image, you can also use any outside paint program. For example, Gimp is a full−featured image manipulation program that is also open−source.

Since a mesh can have layers of UV Textures, there may be many images that color the mesh. However, each UV Texture only has one image.

Using Blender's Texture Paint



The "Paint" tool in action.

Texture Paint works in both a 3D window and the UV/Image Editor window. In the 3D window in Texture Paint mode, you paint directly on the mesh. In the UV/Image Editor window, you paint on a flat canvas that is wrapped around the mesh using UV coordinates. Any changes made in the UV/Image Editor window show up immediately in the 3D window, and vice−versa.

A full complement of brushes and colors can be selected from a floating Image Paint panel in the UV/Image Editor, or a Paint panel in the Buttons window, Editing (F9) context. Brush changes made in either panel are immediately reflected in the other panel.

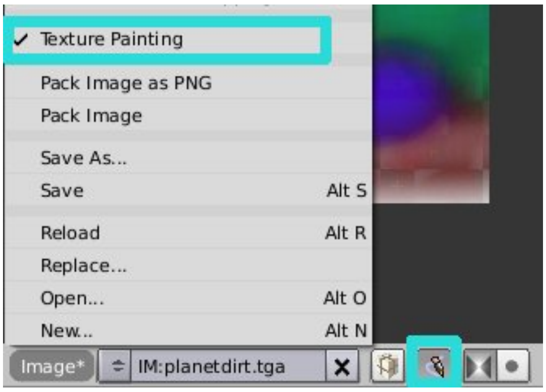
When satisfied or at intermittent intervals, please save your image using the UV/Image Editor window. How to use Texture paint is explained in this section.

Square Power of 2

Texture paint is very fast and responsive when working in the 3D window and when your image is sized as a

square, power of two; 256x256, 512x512, 1024x1024, etc.

Getting Started



Once you have un−wrapped your model to a UV Texture (as explained in previous pages), you have to:

- Load an image into the UV/Image Editor (Image−>Open−>select file) , or
- Create a new image (Image−>New−>specify size) **and** saved it to a file (Image−>Save−>specify file).

You cannot paint on a mesh in Texture Paint mode without **first** unwrapping your mesh, **and** doing one of the above steps. After you have done these two things, you can modify the image using the Texture Paint mode. Once you do:

- In the 3D View window, select Texture Paint mode from the mode selector in the window header, and you can paint directly onto the mesh.
- In the UV/Image Editor window, enable Texture Painting in the Image menu (shown to the right).
- In the UV/Image Editor header, click the magic pencil (highlighted to the right).

At this time, you may choose to show the alpha (transparency) channel by clicking the button to the right of the magic pencil. The dot icon to the right of that button allows you to paint the alpha channel by itself.

Once you enable Texture Painting, your mouse becomes a brush. To work with the UV layout (for example, to move coordinates) you must disable Texture Painting. To work with the mesh in the 3D View (for example, to move it in 3D space), or select other objects, you must leave Texture Paint mode.

When you enable Texture Painting, use the View−>Paint Tool option in the UV/Image Editor window or the Paint panel in the Buttons window to modify paint settings.

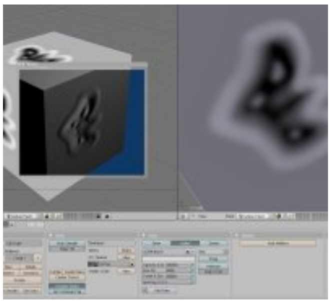
All painting you perform in either window will be instantly reflected in the other window (if the 3D View is in *textured* viewport mode). However, the modified texture will not be saved until you explicitly do so by Image−>Save in the UV/Image Editor window.

See Outline: If you want to paint directly on the mesh in the 3D View window, change to UV/Face Select Mode first to show the edge outline of the mesh, *then* switch to Texture paint mode, which visually overlays the previous mode, but keeps the outline.

As soon as you enable Texture Painting or switch to Texture Paint mode, a Paint Panel becomes available in the Editing (F9) buttons. This panel has all the same controls as those available in the Paint Tool discussed

below. Use this panel if you are only working in 3D view in order to change brushes (colors, patterns, function).

Instant Feedback



If your texture is already used to color, bump map, displace, alpha−transparent, etc. a surface of a model in your scene (in other techie words, is mapped to some aspect of a texture via texture channel using UV as a map input), you can see the effects of your painting in the context of your scene as you paint.

To do this, set up side−by−side windows, one window in 3D View set to Textured display mode, and the second UV/Image Editor window loaded with your image. Position the 3D View to show the object that is UV mapped to the loaded image. Open a Preview window (see 3D View Options for more info) and position it over the object. In the image to the right, the texture being painted is mapped to "Nor" or Normal, and is called "bump mapping", where the gray scale makes the flat surface appear bumpy. See Texture Mapping Output for more information on bump mapping.

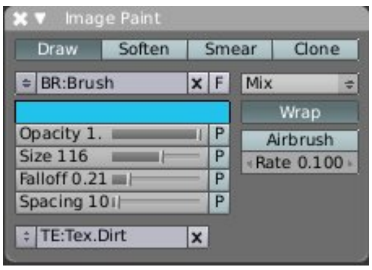
Brushes

To paint on the mesh, you just click and drag the left mouse button across the mesh in the 3D View window, or across the image in the UV/Image Editor window. What that mouse action does to the color of the mesh depends on the Brush you are using. Some brushes add color, some take it away, and some even make your mesh transparent!

Mode: UV/Image Editor

Hotkey: C for Color

Menu: View −> Paint Tool...



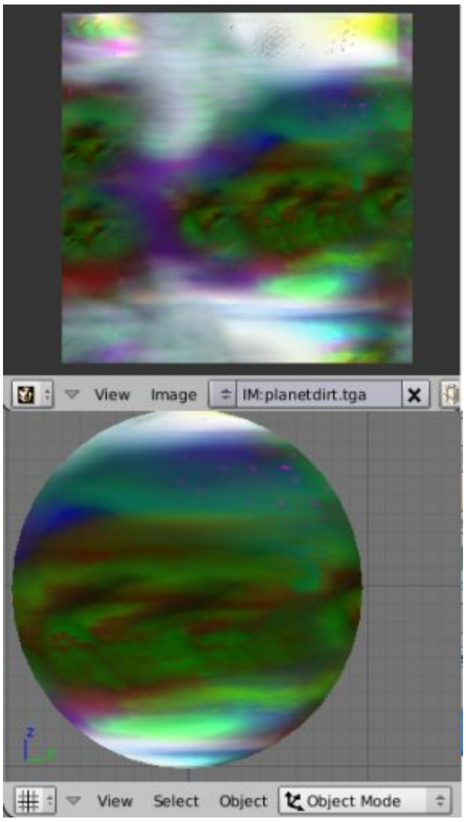
Press C in the UV/Image Editor to pop up the Image Paint panel. With this panel, you can create many brushes, each with unique settings (such as color and width). Use the Brush selector to switch between brushes, or create a new brush. When you add a brush, the new brush is a clone of the current one. You would then change the setting for the new brush. Texture paint has an unlimited number of brushes, and unique

user−defined controls for those brushes, set in the Paint Tool panel. You can paint in either window; you can paint right on your mesh in the 3D window, much like vertex painting, or flat on the UV Image in the UV/Image Editor window.

To use a brush, click on its name from the BR: selector up/down arrow. Name your brush by clicking on the name field and entering any name you wish, such as "Red Air" for a red airbrush. To toss out a brush, click the brush delete X button next to its name. If you want to keep this brush around for the next time you run Blender, click the Fake user button next to the brush delete X button.

If you have a tablet pen with pressure sensitivity, toggle the small "P" button next to the opacity, size, falloff and spacing buttons to control these parameters using the pressure of the pen. Using your pen's eraser end will toggle on the Erase Alpha mode.

Brush Mode Options



Planet texture example using various brush modes

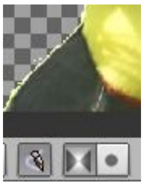
Across the top and down the side are several controls that determine what the brush does when it draws:

- Draw: The normal brush; paints a swath of color
- Soften: blends edges between two colors
- Smear: when you click, takes the colors under the cursor, and blends them in the direction you move the mouse. Similar to the "smudge" tool of *The Gimp*.
- Clone: copies the colors from the image specified (Tex.Dirt in the example), to the active image. The background image is shown when this brush is selected; use the Blend slider to control how prominent the background image is.
- Wrap: wraps your paint to the other side of the image as your brush moves off the OTHER side of the canvas (any side, top/bottom, left/right). Very handy for making seamless textures.

Mix Mode Options

Mix Mode determines what the brush action does to the image:

- Mix: the brush color is mixed in with existing colors
- Add: the brush color is added to the existing color; green added to red gives yellow.
- Subtract: brush color is subtracted; painting blue on purple gives red
- Multiply: the RGB value of the base is multiplied by the brush color
- Lighten: the RGB value base value color is increased by the brush color
- Darken: tones down the colors
- Erase Alpha: makes the image transparent where painted, allowing background colors and lower−level textures to show through. As you 'paint', the false checkerboard background will be revealed.
- Add Alpha: makes the image more opaque where painted



In order to see the effects of the Erase and Add Alpha mix modes in the UV/Image Editor, you must enable alpha channel display by clicking the Display Alpha or the Alpha−Only button. Transparent (no alpha) areas will show an checker background; the clarity of the checker background depends on the opacity of the image.

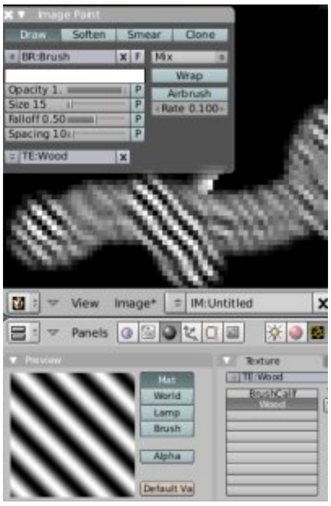
Brush Control

The controls for each brush allow you to choose:

- Color: The current brush color shown in the color swatch. Click the swatch to choose a color for the current brush; then pick a new color from the popup color picker applet. You can pick any hue and then a saturation/value, pre−defined colors, or even use the eyedropper to take a sample from anywhere within the Blender window.
- Opacity: how thickly you apply the paint when you click down, like how hard you press down on a spray can button, or how much paint you put on your brush.
- Size: How big your brush or spray area is, in pixels.
- Falloff: How spread−out the spray is. Decrease for a softer brush; increase for a harder one.
- Spacing: The distance between paint spurts when dragging the mouse, relative to the size of the brush.
- Airbrush: When enabled, the brush paints continually while the LMB is held down, rather than only when the mouse is moving.
- Rate: How fast the paint is applied by the Airbrush

RMB click on any part of the image to sample that color and set it as the brush color.

Brush Pattern/Texture



Use the texture selector at the bottom of the paint panel to select a pre−loaded image or procedural texture to use as your brush pattern. Note that in order to use it, you must have a placeholder material defined, and that particular texture defined using the Material and Texture buttons. It is not necessary to have that material or texture applied to any mesh anywhere; it must only be defined. The example to the right shows the effects of painting with a flat (banded) wood texture. Switching the texture to Rings makes a target/flower type of brush painting pattern.

Note: In Clone paint mode, this field changes to indicate the picture image or texture that you are cloning from.

Saving

If the header menu item Image has an asterisk next to it, it means that the image has been changed, but not saved. Use the Image−>Save Image option in the to save your work with a different name or overwrite the original image.

UV images

Since images used as UV Textures are functionally different from other images, you should keep them in a directory separate from other images.

The image format for saving is independent of the format for rendering. The format for saving a UV image is selected in the header of the Save Image window, and defaults to Targa (.TGA).

If Packing is enabled in the window header, or if you manually Image−>Pack Image, saving your images to a separate file is not necessary.

Options

The Paint controls are also shown on a Paint panel in the buttons window, Editing set.

Mode: Texture Paint

Hotkey: F for Face−select submode

Face Painting: By default, you will be painting the whole mesh and it will look solid. RMB clicking will

pick up the color from the mesh at that point. To only paint part of the mesh, press F in 3D View to highlight the UV/Face edges. The faces of the mesh will be outlined for you using the red−green edges in the 3D View, and if you have Shadow Mesh enabled in the UV/Image Editor, the UV Layout will also be shown as a gray outline in the UV/Image Editor window. Now, RMB will select a face, and Shift RMB will select multiple faces. To only paint part of the mesh, press H to Hide the selected Faces when in Texture Paint mode. Just like when editing a mesh, doing a Border select with the LMB button selects only certain faces. Doing a Border select with the RMB button excludes those faces from the selected set of faces.

Selecting some faces and pressing H Hides them from view and thus painting. It also continues to hide them during textured viewport shading in Object view. During your painting session with F active, or in UV Face Select Mode, Alt H un−hides them. Unfortunately, H and Alt H don't do anything when you are not in this face−select sub−mode (because there aren't faces 'selected').

Using an External Image Editor

If you use an external program to edit your UV Texture, you must:

1. run that paint program (Gimp, Photoshop, Z−Brush, etc.)
2. load the image or create a new one
3. change the image, and
4. re−save it within that program.
5. Back in Blender, you reload the image in the UV/Image Editor window.

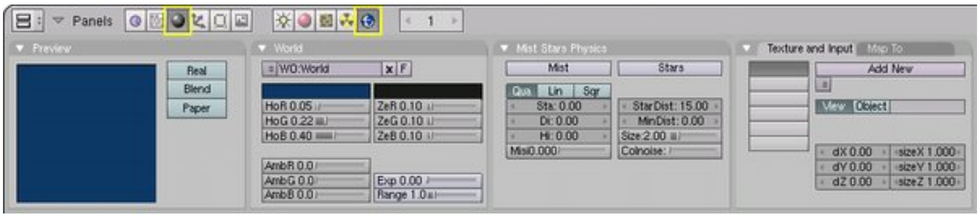
You want to use an external program if you have teams of people using different programs that are developing the UV textures, or if you want to apply any special effects that Texture Paint does not feature, or if you are much more familiar with your favorite paint program.

Previous: Manual/Using UV Textures	Contents	*Next:* Manual/Worlds and Backgrounds

User Manual: Contents | Guidelines | Blender Version **2.43**

THE WORLD AND THE UNIVERSE

Blender provides a number of very interesting settings to complete your renderings by adding a nice background, and some interesting 'depth' effects. These are accessible via the Shading Context (F5) and World Buttons sub−context ( ) shown in *World Buttons*. By default a very plain uniform world is present. You can edit it or add a new World.



World Buttons

Chapters

Previous: Manual/Texture Paint | Contents | *Next:* Manual/World Background

User Manual: [Contents] | [Guidelines] | Blender Version **2.4x**

World Background

Mode: All Modes

Panel: Shading/World Context Preview

Hotkey: F8

Description

The world buttons let you set up the shading of your scene in general. It can provide ambient colour, and special effects such as mist, but a very common use of a World is to shade a background colour.

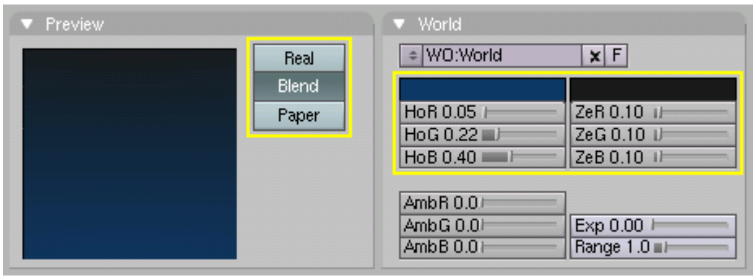
Background Image in Render

To use an image as your render background, see BackBuf images specified in the Output Panel

Background Image in 3D

To use an image as a background image in your 3D view, for example as a reference when doing a model, see using a Background Image

Options



Background colors

HoR, HoG, HoB
: The RGB color at the horizon

ZeR, ZeG, ZeB
: The RGB color at the zenith (overhead)

These colors are interpreted differently, according to the Buttons in the Preview Panel (*Background colors*):

Blend
: The background color is blended from horizon to zenith. If only this button is pressed, the gradient runs from the bottom to the top of the rendered image regardless of the camera orientation.

Real
: If this option is added, the gradient produced has two transitions, from nadir (same color as zenith) to horizon to zenith; the blending is also dependent on the camera orientation, witch makes it more

realistic. The horizon color is exactly at the horizon (on the x−y plane), and the zenith color is used for points vertically above and below the camera.

Paper
: If this option is added, the gradient keeps its characteristics, but it is clipped in the image (it stays on a horizontal plane (parallel to x−y plane): what ever the angle of the camera may be, the horizon is always at the middle of the image).

Textures

Instead of a color, or blend of two colors, Blender can use an 2D image which it maps to a very large Box or sphere which encompasses the entire scene, or which it maps to a virtual space around the scene.

Texture buttons

The World Buttons also provide a stack of texture channels. They are used much like the Materials textures, except for a couple of differences (*Texture buttons*). The textures can be mapped according to:

View
: The default orientation, aligned with the co−ordinates of the final render

AngMap
: Used to wrap a standard hemisphere angular map around the scene in a dome. This can be used for image based lighting with Ambient Occlusion set to sky color. You'll generally need a high dynamic range image (HDRI) angular map (the look like a weird spherical image).

Sphere
: Sphere mapping, similar to that of materials

Tube
: Wrap the rectangular texture around in a cylinder, similar to that of materials

Object
: Position the texture relative to a specified object's local texture space

The texture affects color only, but in four different ways:

Blend
: Makes the Horizon color appear where the texture is non−zero

Hori
: Affect the color of the horizon

ZenUp
: Affect the zenith color overhead

ZenDo
: Affect the zenith color underneath

Previous: Manual/Worlds and Backgrounds	Contents	*Next:* Manual/Ambient Occlusion

User Manual: Contents | Guidelines | Blender Version **2.44**

Ambient Occlusion

Mode: All Modes

Panel: Shading/World Context Amb Occ

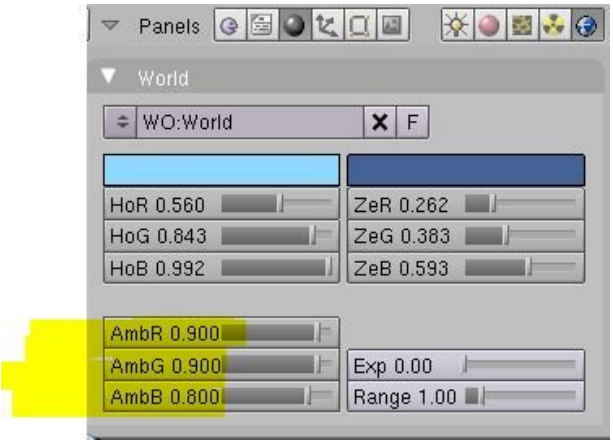
Hotkey: F8

Description

Ambient Occlusion is a sophisticated raytracing calculation which simulates soft global illumination by faking darkness perceived in corners and at mesh intersections, creases, and cracks, where light is diffused (usually) by accumulated dirt and dust. This has the effect of darkening cracks, corners and points of contact, which is why Ambient Occlusion is often referred to as a 'dirt shader'.

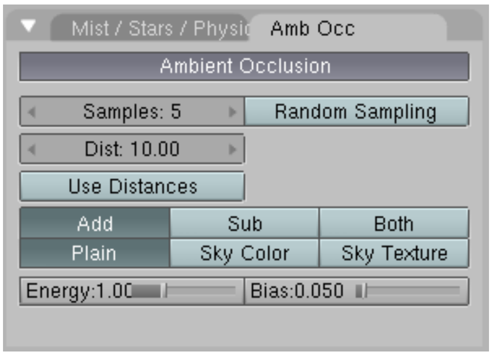
There is no such thing as AO in real life, AO is a specific not−physically−accurate (but generally nice looking) rendering trick. It basically samples a hemisphere around each point on the face, sees what proportion of that hemisphere is occluded by other geometry, and shades the pixel accordingly. It's got nothing to do with light at all, it's purely a rendering trick that tends to look nice because generally in real life surfaces that are close together (like small cracks) will be darker than surfaces that don't have anything in front of them, because of shadows, dirt, etc. The AO process though is approximating this result, it's not simulating light bouncing around or going through things. That's why AO still works when you don't have any lights in the scene, and it's why just switching on AO alone is a very bad way of 'lighting' a scene.

You must have Raytracing enabled as a Render panel option for this to work.



You must have an ambient light color (World settings, AmbRGB sliders) set to your desires. By default, the ambient light color (world) is black, simulating midnight in the basement during a power outage. Applying that color as ambient will actually darken all colors. A good outdoor mid−day color is RGB (0.0,0.9,0.8) which is a whitish yellow sunny kind of color on a bright but not harshly bright, day.

Options

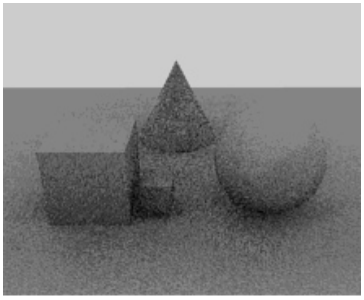


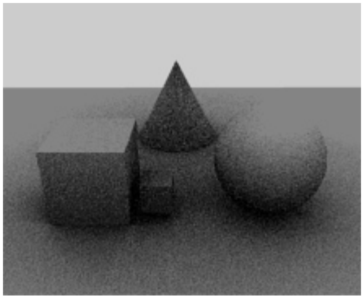
Ambient Occlusion Panel.

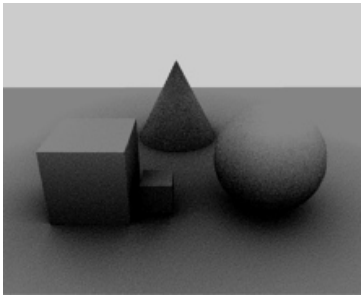
Sampling

Samples

The number of rays used to detect if an object is occluded. Higher numbers of samples give smoother and more accurate results, at the expense of slower render times. The default value of 5 is usually good for previews. The actual amount of shot rays is the square of this number. (i.e. Samples=5 means 25 rays). Rays are shot at the hemisphere according to a random pattern, this causes differences in the occlusion pattern of neighbouring pixels unless the number of shot rays is big enough to produce good statistical data.


Ambient Occlusion with 3 Samples


Ambient Occlusion with 6 Samples


Ambient Occlusion with 12 Samples

Dist

The length of the occlusion rays. The longer this distance, the greater impact that far away geometry will have on the occlusion effect. A high Dist value also means that the renderer has to search a greater area for geometry that occludes, so render time can be optimized by making this distance as short as possible, for the visual effect that you want.

Use Distances, DistF

Controls the attenuation of the shadows. Higher values give a shorter shadow, as it falls off more quickly.

Blending

The ambient occlusion pass is composited during the render pipeline. Three blending modes are available:

Add

The pixel receives light according to the number of non−obstructed rays. The scene is lighter.

Sub

The pixel receives shadow (negative light) according to the number of obstructed rays. The scene is darker.

Both
: Both effects take place, the scene has more or less the same brightness, but more contrast.

Energy
: The strength of the AO effect, a multiplier for addition or subtraction

Note

If **Sub** is chosen, there must be other light sources, otherwise the scene will be pitch black. In the other two cases the scene is lit even if no explicit light is present, just from the AO effect. Although many people like to use AO alone as a quick shortcut to light a scene, the results it gives will be muted and flat, like an overcast day. In most cases, it is best to light a scene properly with Blender's standard lamps, then use AO on top of that, set to 'Sub', for the additional details and contact shadows.

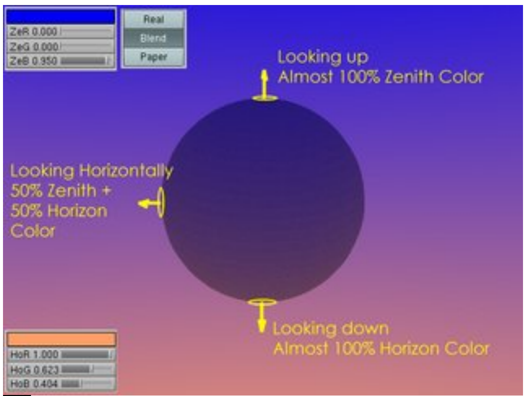
Ambient Color

Ambient Occlusion can take the color of its lighting from various sources

Plain
: The pixel receives shading based on the World's ambient color

Sky Color
: The pixel receives shading based on the World's sky color. The color is computed on the basis of the portion of the sky hit by the non−obstructed rays (*Ambient Occlusion with Sky Color. Zenith is blue, Horizon is orange, and type is Blend so that sky goes full orange at Nadir.*).

Sky Texture
: A Sky Image texture must be present, possibly an AngMap or a SphereMap. It behaves as Sky Color but the ray color depends on the color of the Sky texture pixel hit.



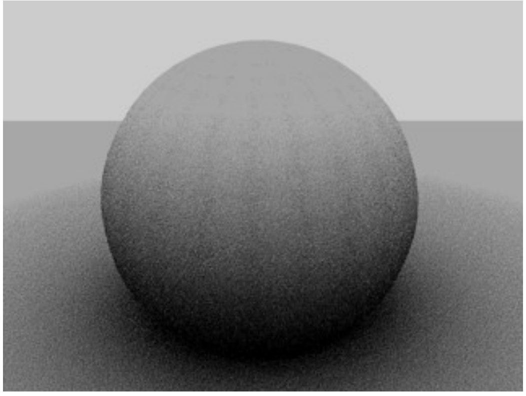
Ambient Occlusion with Sky Color. Zenith is blue, Horizon is orange, and type is Blend so that sky goes full orange at Nadir.

Ambient Occlusion with Sky Texture, using a scaled down St. Peters Basilica HDR AngMap [1]

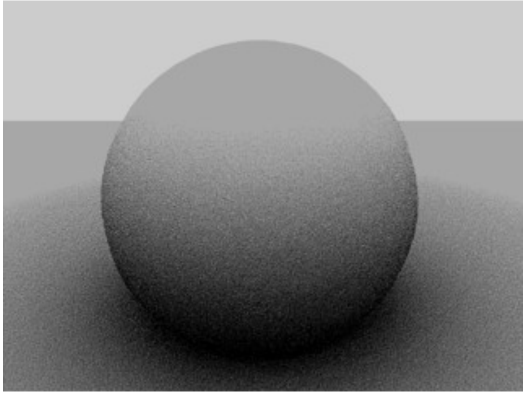
Bias

Bias
: The angle (in radians) the hemisphere will be made narrower.

The bias setting allows you to control how smooth 'smooth' faces will appear in AO rendering. Since AO occurs on the original faceted mesh, it is possible that the AO light makes faces visible even on objects with 'smooth' on. This is due to the way AO rays are shot, and can be controlled with the Bias Slider.



24x24 UV Sphere with Bias: 0.05 (default). Note the facets on the sphere's surface even though it is set smooth



Raising the Bias to 0.15 removes the faceted artifacts

Technical Details

Ambient Occlusion is calculated by casting rays from each visible point, and by counting how many of them actually reach the sky, and how many, on the other hand, are obstructed by objects.

The amount of light on the point is then proportional to the number of rays which have 'escaped' and have reached the sky. This is done by firing a hemisphere of shadow−rays around. If a ray hits another face (it is occluded) then that ray is considered 'shadow', otherwise it is considered 'light'. The ratio between 'shadow' and 'light' rays defines how bright a given pixel is.

Hints

Ambient Occlusion is a raytracing technique, so it tends to be slow. Furthermore, performance severely depends on Octree size, see the Rendering Chapter for more information.

Previous: Manual/World Background	Contents	*Next:* Manual/Exposure

User Manual: Contents | Guidelines | Blender Version **2.4x**

Exposure and Range

Mode: All Modes

Panel: Shading/World Context Wold

Hotkey: F8

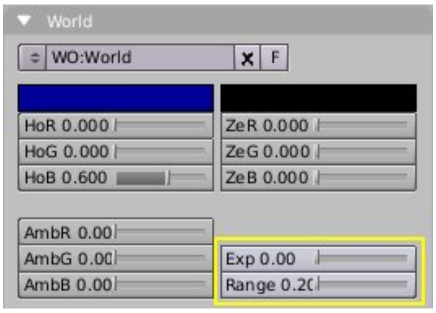
Description

Exposure and Range are similar to the "Colour Curves" Tool in Gimp or Photoshop.

Previously Blender clipped color straight with '1.0' (or 255) when it exceeded the possible RGB space. This caused ugly banding and overblown highlights when light overflowed (*An overexposed Teapot*).

Using an exponential correction formula, this now can be nicely corrected.

Options



Exposure and Range Buttons

Exp
: The exponential curvature, with 0.0 being linear, and 1.0 being curved.

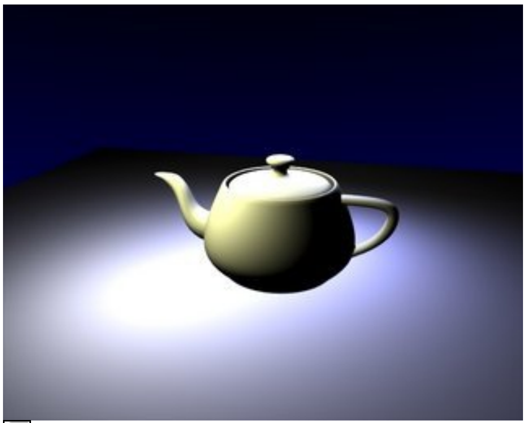
Range
: The range of input colors that are mapped to visible colors (0.0−1.0).

So without Exposure we will get a linear correction of all colour values:

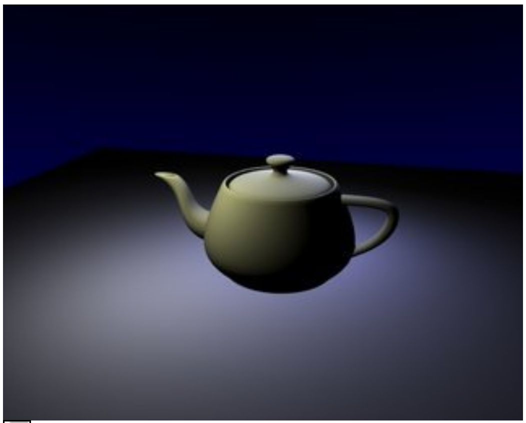
1. Range > 1.0: the picture will become darker; with Range = 2.0, a color value of 1.0 (the brightest by default) will be clipped to 0.5 (half bright). (*Range 2.0*).
2. Range < 1.0: the picture will become brighter; with Range = 0.5, a color value of 0.5 (half bright by default) will be clipped to 1.0 (the brightest). (*Range 0.5*).

Examples

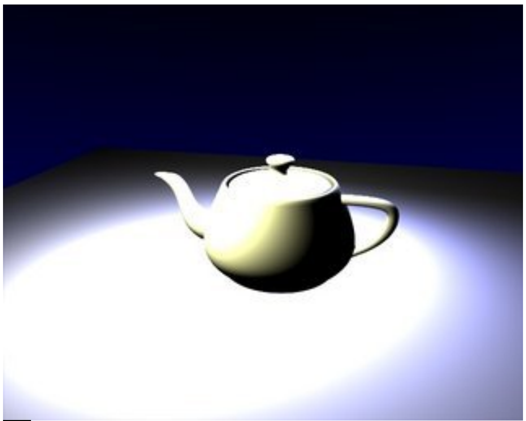
With a linear correction every colour value will get changed, which is probably not what we want. Exposure brightens the darker pixels, so that the darker parts of the Image won't be changed at all (*Range 2.0*, *Exposure 0.3*).



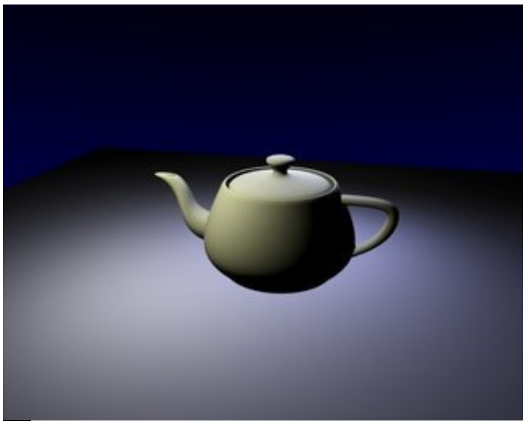
An overexposed Teapot.



Range 2.0



Range 0.5



Range 2.0, Exposure 0.3

Hints

Try and find the best Range value, so that overexposed parts are just not too bright. Now turn up the Exposure value, until the overall brightness of the image is satisfying. This is especially usefull with area lamps.

Previous: Manual/Ambient Occlusion	Contents	*Next:* Manual/Mist

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.4x**

Mist

Mode: All Modes

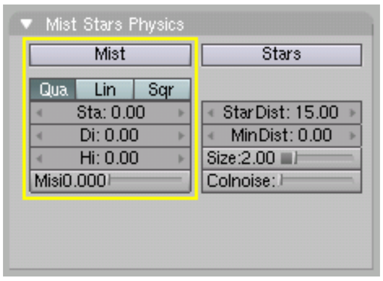
Panel: Shading/World Context Mist Stars Physics

Hotkey: F8

Description

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender makes objects farther away more transparent (decreasing their Alpha value) so that they mix more of the the background color with the object color. With Mist enabled, the further the object is away from the camera the less it's alpha value will be.

Option



Mist buttons

Mist
: Toggles mist on and off.

Qua / Lin / Sqr
: The decay rate of the mist, Quadratic, Linear, and Square Root. These settings the rate of change of the mist's strength further and further into the distance.

Sta
: The distance from the camera at which the mist starts to fade in

Di

The distance from the Start of the mist, that it fades in over. Objects further from the camera than Sta+Di are completely hidden by the mist.

Mist distances

To visualize the mist distances in the 3D View, select your camera, go to Editing Context (F9) and enable *Show Mist* in the *Camera* Panel.

The camera will show mist limits as a line projecting from the camera starting from *Sta* and of distance *Di*.

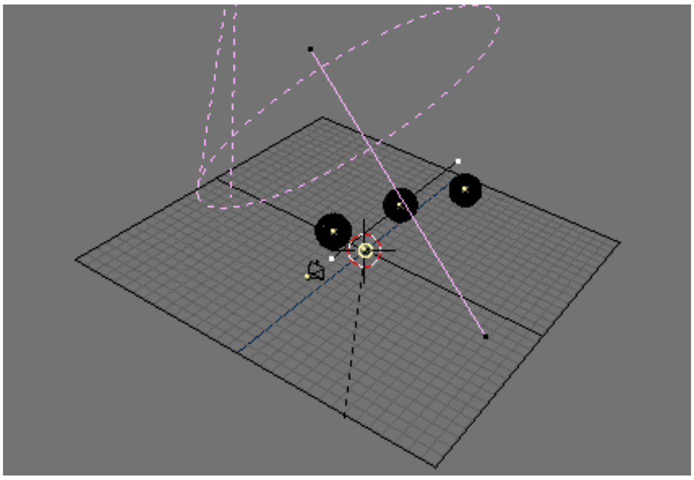
Hi

Makes the mist intensity decrease with height, for a more realistic effect. If greater than 0, it sets, in Blender units, an interval around z=0 in which the mist goes from maximum intensity (below) to zero (above).

Misi
: An overall intensity, or strength, of the mist.

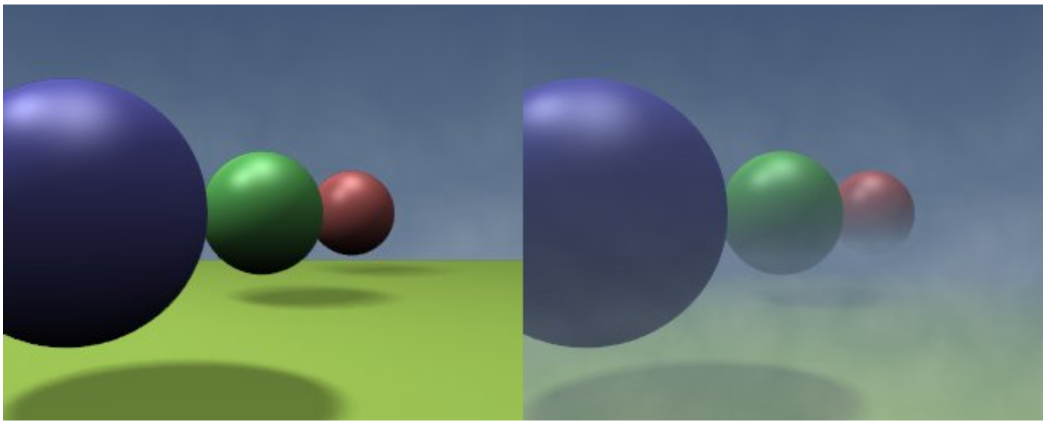
Examples

Mist test setup shows a possible test set up.



Mist test setup

Rendering without mist (left) and with mist (right). shows the results with and without mist. The texture is a plain procedural cloud texture with Hard noise set.



Rendering without mist (left) and with mist (right).

User Manual: Contents | Guidelines | Blender Version **2.4x**

Stars

Mode: All Modes

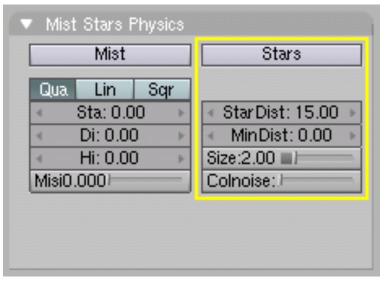
Panel: Shading/World Context Mist Stars Physics

Hotkey: F8

Description

Stars are randomly placed halo−like objects which appear in the background. The Stars settings are on the right−hand side of the Mist Stars Physics Panel (*Star buttons*).

Options



Star buttons

StarDist
: The *average* distance between stars. Stars are intrinsically a 3D feature, they are placed in space, not on the image!

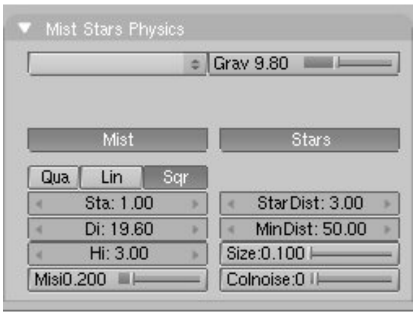
MinDist
: The *minimum* distance from the camera at which stars are placed. This should be greater than the distance from the camera to the *furthest* object in your scene, unless you want to risk having stars *in front* of your objects.

Size
: The actual size of the star halo. It is better to keep it much smaller than the proposed default, to keep the material smaller than pixel−size and have pin−point stars. This is much more realistic.

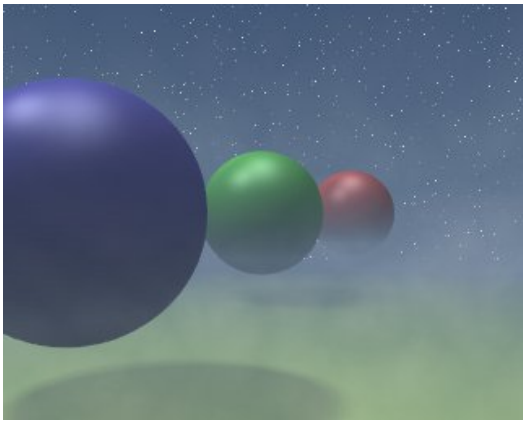
Colnoise
: Adds a random hue to the otherwise plain white stars. It is usually a good idea to add a little ColNoise.

Examples

Star rendering. shows the same misty image from the Mist chapter but with stars added. The Stars settings used for the image are shown in *Star settings*..



Star settings.



Star rendering.

Previous: [Manual/Mist]()	[Contents]()	*Next:* [Manual/Physics]()

Physics Engine

Mode: All Modes

Panel: Shading/World Context Mist Stars Physics

Hotkey: F8

Description

This part of the World determines the type of physical rules that govern the **game engine** scene, and the gravity of the situation (sorry, couldn't resist the pun). Based on the physics engine selected, in physics simulations in the game engine, Blender will automatically move Actors in the downward (−Z) direction. After you arrange the Actors and they move as you wish, you can then bake this computed motion into fixed IPO curves (see Logic−>Actor−>Dynamic for more info).

Options

Physics
: The type of physics engine to use:

  Bullet
  : The default physics engine, in active development. It handles movement and collision detection. The things that collide transfer momentum to the collided object.

  Sumo (Deprecated)
  : An older physics engine, used in past versions of Blender's game engine. Use this for compatibility with old files.

  None
  : No physics in use. Things are not affected by gravity and can fly about in a virtual space. Objects in motion stay in that motion.

Gravity
: The gravitational acceleration, in units of meters/second/second, of this world. Each object that is an actor has a mass and size slider (see Materials section). In conjunction with the frame rate (see Render section), Blender uses this info to calculate how fast the object should accelerate downward.

Previous: Manual/Stars	Contents	*Next:* Manual/Animation Basics

User Manual: Contents | Guidelines | Blender Version **2.43**

Animation is making an object move or change shape over time. Objects can be animated in many ways. They can be animated as one or more simultaneous ways:

Moving as a whole object
: Changing their position, orientation or size in time;

Deforming them
: animating their vertices or control points;

Character Animation via Armature
: animated to deform by the movement of bones inside the mesh, a very complex and flexible interaction that makes character−shaped objects appear to walk and jump.

In this chapter we will cover the first case, but the basics given here are actually vital for understanding the following chapters as well.

Three methods are normally used in animation software to make a 3D object move:

Key frames
: Complete positions are saved for units of time (frames). An animation is created by interpolating an object fluidly through the frames. The advantage of this method is that it allows you to work with clearly visualized units. The animator can work from one position to the next and can change previously created positions, or move them in time.

Motion Curves
: Curves can be drawn for each XYZ component for location, rotation, and size. These form the graphs for the movement, with time set out horizontally and the value set out vertically. The advantage of this method is that it gives you precise control over the results of the movement.

Path
: A curve is drawn in 3D space, and the Object is constrained to follow it according to a given time function of the position along the path.

The first two systems in Blender are completely integrated in a single one, the *IPO* (InterPOlation) system. Fundamentally, the IPO system consists of standard motion curves. A simple press of a button changes the IPO to a key system, without conversion, and with no change to the results. The user can work any way he chooses to with the keys, switching to motion curves and back again, in whatever way produces the best result or satisfies the user's preferences. The IPO system also has relevant implication in Path animations.

Chapters

See also Manual/Hooks − Uses a modifier as a way to change the shape of a mesh. Sorta like sticking a fish hook in a mesh and pulling. Uses the principles discussed in Shape Keys.

Previous: Manual/Stars	Contents	*Next:* Manual/Ipo Types

User Manual: Contents | Guidelines | Blender Version **2.46**

Interpolation (Ipo) is the process of estimating an object's position (or other attributes) based on a known start and end value, and the time between the start and the end. If I stand in 3D space (0,0,0) at frame 1, and I want to move to (2,2,0) by frame 10, then Blender can interpolate, for frame 5, that I will be half−way there, or at (1,1,0). To make that interpolation, you define a bezier curve which connects the two points, and Blender uses the curve to figure out the intermediate values.

Interpolation in Blender

Mode: All Modes

Hotkey: I

Menu: Object −> Insert Keyframe

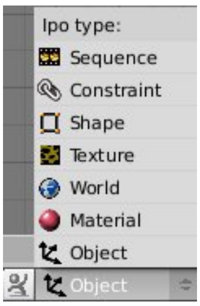
The IPO datablock in Blender is universal. It makes no difference whether an object's movement is controlled, or its material settings, or its size, etc. Each of these aspects is called a *channel*. There is a channel for the object's X location, a channel for its Y location and so on. Once you have learned to work with object IPOs, how you work with other channels will become obvious. Blender does distinguish between different *types* of IPOs and the channels that are available, and keeps track of all of that automatically.

Every type of IPO block has a fixed number of available *channels*. These each have a name (LocX, SizeZ, etc.) that indicates how they are applied. When you add an IPO Curve to a channel, animation begins immediately. At your discretion (and there are separate channels for this), a curve can be linked directly to a value (LocX...), or it can, in some instances, affect a variance of it (dLocX...). The latter enables you to move an object as you would usually do, with the Grabber, without disrupting the IPO. The actual location is then determined by IPO Curves *relative* to that location.

The Blender interface offers many options for copying IPOs, linking IPOs to more than one object (one IPO can animate multiple objects.), or deleting IPO links. The IPO Window Reference section gives a detailed description of how to work with curves and their handles.

If there is an IPO curve for a channel, the interpolated value overrides whatever panel setting is changed by you. You can change the property value through the panel and re−key (through the I−key command) to take on that value for that frame. Otherwise, it will reset to the interpolated curve value when you render or change frames! This can be very confusing if you forget that a channel has been automated. You can always select a channel, select the curve, and Delete X it if you do not need it to be animated.

Ipo Curve Types



Ipo Types menu

As well as the default Object Ipo, there are Ipos to control animation of Materials, Worlds, Shapes, Constraints, Video Sequence strips and more. For every kind of object in Blender, you can animate (change over time):

- Object − the object itself as a whole, such as its location, rotation, layer...
- World − World environment settings, such as horizon color, ambient light...
- Constraint − limits placed on the object
- Sequence − a video effect strip in the Video Sequence Editor.

Ipo Types menu

Depending on the kind of object that is selected, additional animatable aspects become available. For example, the Curve Ipo type appears if the selected object is a curve and not a mesh; the Lamp Ipo type only appears if the selected object is a lamp. Whole other additional channels, or sets of channels become available for animating based on the object selected:

- Mesh − Materials (colors, specularity, alpha), Textures and the Shape of the mesh
- Curve − Shape as well as Path Speed (to vary the speed of an object following the curve as a path)
- Surface − Shape of NURBS surfaces (donuts, spheres) deformed
- Meta − no additional animation
- Text − no additional animation
- Empty − nothing additional to animate
- Camera − Lens, clipping boundaries, Aperture, Focal Distance...
- Lamp − energy (brightness), color, texture...
- Armature − Poses, which are rotation sets for all the bones in the armature
- Lattice − Shape of the Lattice

On the IPO window header, after the menu, there is an eye icon. Enable this to **mute** (or blind Blender to) the channel and any IPO curves. This is handy when debugging your scene in trying to determine, for example, whether lighting or material changes are making your character look funny.

Next over is the IPO Type, and you can click the up/down selector to change between the various IPO Types. For those channels that map to a texture, there will be a texture channel number selector that indicates which texture channel mapping is being automated.

Next over is the name of the IPO datablock. Different materials can share the same datablock. For example, you can make all materials fade out by creating an IPO curve of just the Alpha channel. If you applied that curve to each material in your scene, then regardless of the material's color, spec, etc, they all would fade out as their alpha channel was controlled by the curve. Click the X to delete, or the F to Fake user and retain this curve, whether or not anyone actually uses it.

Next over are the curve copy/paste arrow buttons, the zoom selector (click it and then box select a range of frames to focus on), and the lock icon which forces other windows to update when you make changes.

Animating Materials

Material Ipos can be used for animating a Material so that it changes over time; for example, a character's face can turn from red to blue in 10 seconds as he holds his breath. Just as with objects, IPO Material curves can be used to specify 'key positions' for Materials. Each aspect of a material is animated through a channel. Just as a Object IPO has channels for automating Location X, Location Y, etc, the Material IPO has channels for Color Red, Color Green, etc.

Any animation you do for a material will be reflected in all objects (and their material index) that share that material. Any animation can be driven by some aspect of another object (see IPO Drivers). Once a material channel is animated, the panel sliders and values reflect what the value is for that frame. Changing the value in the panel will not have any effect, because the value is being determined by the interpolated curve.

To access these channels in the IPO window, select Material as the Ipo type. There may be more channels than can be displayed in the window. To see channels that have scrolled off the screen, drag your MMB to pan the window up and down. Pan the workspace to see the curve (or work in Keyframe mode.)

Creating Channels

There are dozens of Material channels. For memory and processing efficiency, and to just help your workflow efficiency, when you first create an IPO for a material, you get to choose a set of commonly used materials channels. With the mouse in the Buttons Window, displaying the Shading F5 Materials buttons, the command I calls up a pop−up menu with options for various sets of Material channels:

RGB
: Color of the material

Alpha
: Transparency of the mesh/material

Halo Size
: Diameter of the Halo, if selected

Mode
: An integer that specifies different settings of the material
    - ◊ 1 − Traceable for Ray−tracing, but does not receive shadows
    - ◊ 2 − Not traceable, but receives shadows
    - ◊ 3 − Shadeless
    - ◊ and so on. The easiest way to determine the rest is to play.

All Color
: Many color−related channels − base, specular, alpha, emit (16 in all)

All Mirror
: Many ray−traced reflectivity variables − values and color

Ofs
: The offset of the selected texture to the surface

Size
: The dimensions of the texture applied to the mesh/material

All Mapping
: all the texture settings − color (actual color and amount of influence, Normal amount, offsets, size and so on.

The above−described popup list is just a commonly used set of channels, and not all possible channels are included in the superset of those lists. e.g., the RGB option just initializes/sets a curve for the three R, G, and B channels. For Emit, the "All Color" selection keys that channel among others.

The complete exhaustive list of Material channels is listed down the side of the IPO Window. Choosing a selection from the popup menu may, but not necessarily, set curves and values for every related IPO channel. For example, you can go through all the popup menu selections, and yet Mirror color is not keyed, even though you would think it should be when you select "All Mirror".

Consider memory and processing efficiency when selecting a set. It's inefficient to create a curve (and then have to interpolate it for every frame) when chances are it is not employed/utilized to actually do anything. The Material sets are there to keep the number of channels that Blender has to sift through to a minimum. Remember that when rendering, Blender has to go through every object x materials x channels to determine a value, so more channels is a multiplier effect on render time.

Animating Channel Values

After you select a set, IPO curves are created for the channels in that set, and the current values for those channels are locked in. Not all values for all channels are locked in. The channel may be listed, but not have a curve initialized. There are two ways to initialize a curve for a selected channel. First, LMB select the channel (for example, Emit). Then, in the IPO window, either:

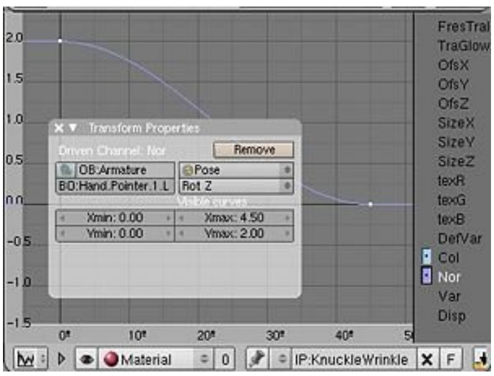
- Ctrl LMB click anywhere in the window itself to insert a control point where the cursor is. So if you want emit to be 0 initially, click at (frame 1, value 0). Blender creates a constant value (horizontally flat) curve. Move the mouse/scroll the window to the frame where you want it to change, and click again. Blender creates a curve, which you can then change.
- select another channel and copy that curve to the buffer in the IPO Editor (use the little down arrow on the header). Then select the target channel and apply it to the desired selected channel (up arrow button).

Material curves are like any other curve type, and can be different kinds of control points, curve types, extend modes, and can be displayed as Curves or Keyframes.

Animating Texture Channel Mapping

If you are in a Material, Lamp or World Ipo Block, then a small Number field appears next to the IPO type Menu in the IPO Window toolbar. This indicates which texture channel is active. Recall that each Material can Map Input and Map To a texture, and these setting can be animated. The mapping for all texture channels can be controlled with Ipo Curves. Click the number and enter a number to change the texture channel being animated. Channel 0 is the top channel or base texture, and goes from there (channel 1 is the next channel down in the list). The Texture itself can be animated, but that is a different IPO Curve Type.

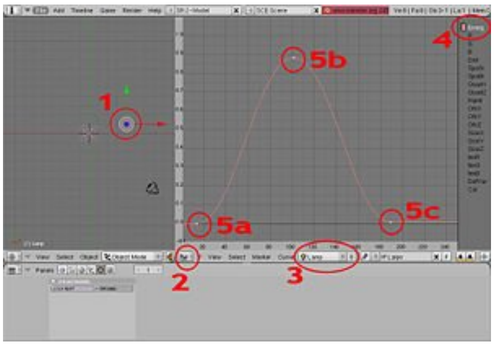
Driving Channel Values



In addition to animating a channel, such as a Material's Color, manually for specific frames, you can drive the change by having some other object do something, like rotate on its Z axis. As that other driver object rotates, blender changes the material color. This relationship is called a Driver relationship, and it works through an Ipo curve as well. Another example is a dimmer knob that rotates to control the lamp's energy channel. If, in the animation, you animate the knob turning, the energy output of the lamp will go up and down. This one knob can control many lamps; for example rotating it in the Z direction could make the inside lights go brighter and the outside sun get darker. the relationship between the knob turning and the energy changing is controlled through a shaped curve, so you can make the relationship smooth, linear, or abrupt.

In the example to the right, we have a finger mesh with a ridged texture on the top texture channel (channel 0). When the bone (Hand.Pointer.1.L − the first bone of the left hand's pointer finger) is straight (rotation Z is 0), the Normal value of the texture channel is 2.0. As the finger bone bends toward 45 degrees, the Nor value decreases to 0, simulating the skin smoothing out. If the bone continues past 45 degrees, the Nor value remains at 0.

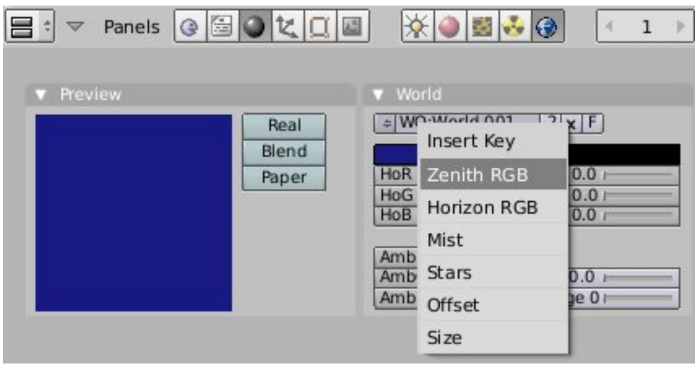
Animating Lamps



A Lamp IPO is yet another type of Ipo that controls aspects/channels of a lamp. You can, for example, change the color of a lamp over time by setting Ipo curves for the red, green and blue channels. The example to the right shows how to animate the energy (brightness) of a lamp:

1: Select the Lamp you want to control via RMB in the 3D View.
2: Make sure you have an IPO window open.
3: Set the IPO type to Lamp.
4 Click on "Energy" to establish a lamp energy IPO.
5a,b,and c: Use Ctrl LMB click to set the points for your IPO curve.

Animating World Settings



Pressing I in the Buttons window while the material World settings are displayed allows you to make an Ipo curve for any of the channel groups shown to the right. Within each channel group are individual channels.

For example, in the Horizon group there are three channels, HorR, HorG, and HorB for Horizon Red, Horizon Green, and Horizon Blue, respectively. You see these channels by selecting the World Ipo Type in the Ipo Window. Setting up an Ipo curve (via setting the color and pressing I in the Buttons window, or by manually creating points via Ctrl LMB clicking in the Ipo window) animates these settings over time. By animating the world settings, you can simulate sunrise to sunset.

Animating Constraints

At the bottom of each Constraint panel is an Influence slider. Next to the slider is a Show and Key button. The Key button creates an Ipo Datablock and keys the influence setting at that frame. You can view the Ipo curve by selecting the Constraint type of curve set. This curve should vary between 0 and 1. A practical example is when you want a camera to track an object for a while, and then perhaps shift attention to a different object. In this example, you would create one constraint, with an influence of 1.0 between frames 10 and 50, and zero at all other times. If you use a bezier curve and key 0 at frame 1, then the camera will swing from its default position to point at the object between frames 1 and 10, track to the object until frame 50, and then resume its normal transformation.

Sharing Ipo Datablocks

Animation curves, such as delta location, can be shared by multiple objects. Defining a relative motion (dLoc) curve for one object would result in an Ipo Datablock that holds that displacement curve. You can easily assign that same curve to another object simply by selecting the object and using the Ipo selector (in the Ipo Window) to choose that Ipo curve. That Ipo datablock will then have 2 users. Each object will, during the same time frame, move relative to their starting position by the same amount.

If you want to delink the selected object from its Ipo curve, click the X next to the Ipo Name. If you want to make some customizations to the motion of the active object only (and not all the other objects that share this Ipo), you have to make a single user copy of the Ipo curve for this object. To do this, click on the Number of Users button (the 2 in the example above) located on the header next to the name. Blender will prompt you to confirm making a single user copy, and if you confirm, a new Ipo will be created (the name will change) and it will be assigned only to the active object.

See Also

- Reference Manual: IPO Window

Previous: Manual/Animation Basics | Contents | *Next:* Manual/Editing Ipos

User Manual: Contents | Guidelines | Blender Version **2.43**

Mode: All Modes

Hotkey: I

Menu: Object −> Insert Keyframe

The simplest method for creating an object IPO is with the "Insert key" (I) command in the 3DWindow, with an Object selected. A Pop−up menu provides a wide selection of options (*Insert Key Menu.*). Selecting an option (eg. Loc) saves the current X/Y/Z location and everything takes place automatically:

- If there is no IPO block, a new one is created and linked to the object.
- If there are no IPOCurves in the channels LocX, LocY and LocZ, these are created.
- Vertices are then added in the IPOCurves with the exact values of the object location.

Example

Position your Scene to Frame 1. Add a mesh object to your Blender Scene. The object is added at the 3D Cursor loctation in edit mode. Press Tab to leave edit mode, and I to Insert a key. In the popup Insert Key window, select Loc. In your Ipo Window, you now see, for the Object mode, the LocX, LocY, and LocZ channels have been filled and selected. There is now a (flat) curve with curve points at Frame 1. The Y value of these points define the location (value) of that channel at that point in time.

Go 30 frames further on (3 x â†‘) and move the object by Grabbing it and dropping it somewhere else. Again use I. Now we can immediately press Enter since Blender remembers our last choice (Loc) and will highlight it. The new position is inserted in the IPO Curves at frame 30.

We can see the animation by slowly paging back through the frames (â†). The object moves between the two positions. In this way, you can create the animation by paging through the frames, position by position. Note that the location of the object is *directly* linked to the curves. When you change frames, the IPOs are always re−evaluated and re−applied. You can freely move the object within the same frame, but as soon as you change frame, the object 'jumps' to the position determined by the IPO.

The rotation and size of the object are completely free in this example. They can be changed or animated with the Insert key procedure by selecting from the I menu the other options such as Rotation, Size and any combination of these.

Recording IPOs

Blender has a built−in physics simulation engine ("Bullet") which allows you to define objects as having a mass, and being solid, and having velocity and angular momentum. Once you set this up, you can use the main menu Game−>Record Game Physics to IPOs" selection to toggle on recording. Now, when you next press P for Play, the actual location and rotation of the selected object will be recorded for each frame of the animation.

Use this feature for recording complex interactions of multiple animated objects. For more information on using the Blender Game Engine, check out the appropriate wiki book from the main wiki page.

Linking IPOs

An Ipo is (in Blender) a datablock, in that it is a separate object. You make it act on an object by linking it to the object. You can offset that link using the TimeOffset in the Anim settings for the object.

Previous: Manual/Ipo Types	Contents	*Next:* Manual/Ipo Curves and Keyframes

User Manual: Contents | Guidelines | Blender Version **2.45**

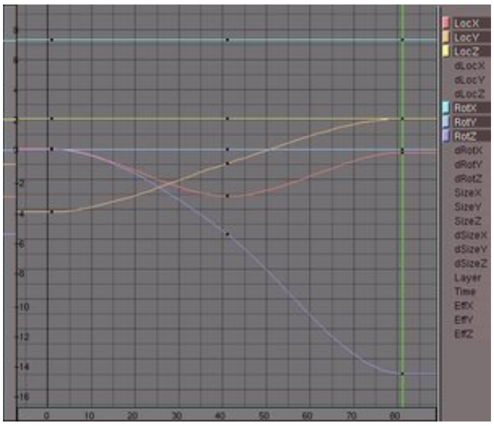
There are two views of your animation, and thus two ways to define and adjust animation: curves and keys. A **curve** connects two points. The shape of the curve is adjusted by moving handles for the points. The handles assert the degree of influence that the point has on the curve. A **key** is a combination of values of points at a particular point in time. Using keys in animation is called Keyframing.

Ipo Curves

Mode: Ipo Curve Editor

Hotkey: C

Menu: View−>Show Keys (to disable)



The Ipo Window − Curve display mode

The Ipo Curve Editor allows you to edit the 2D curves that define animation in Blender. The curves represent the edited value in the vertical (Y) axis (location, size, rotation, energy, etc.) and time on the horizontal (X) axis. The rate of change of these values over time can be seen in the slope of the curve.

You can turn any window into an Ipo Curve Editor with the Window Type menu, but it is often handy to have both a 3D View and an Ipo editor at the same time. This shows all the Ipo Curves, the channels used and those available. You can zoom in and out the Ipo Window and translate it just as every other Blender Window.

In addition to the standard channels, which can be set via I, you have the *delta* options, such as dLocX. These channels allow you to assign a *relative* change. This option is primarily used to control multiple objects with the same Ipo. In addition, it is possible to work in animation 'layers'. To set an Ipo Curve for these channels, select the channel and then create control points via Ctrl LMB clicking in the Ipo Window.

The Ipo curves have an important feature that distinguishes them from normal curves: it is impossible to place more than one curve segment horizontally. Loops and circles in an Ipo are senseless and ambiguous. An Ipo can only have 1 value at a time. This is automatically detected and corrected in the Ipo Editor. Every Ipo channel that has already been keyed is shown in the channel list to the right of the window with a color button by its side.

Curve showing/hiding

You can show an Ipo curve by using LMB on the relative Ipo channel name on the right (one that already possess a color button next to list channel name). By doing this you hide all previously shown Ipo curves. If you want to show many Ipo curves together (for example if you're working with a group of channels such as the LocX, LocY, LocZ) use shift LMB on every channel name you want to be shown (their names appear in white).

Curve Selection

Mode: All Modes

Menu: Ipo Curve Editor Select

Some of the standard Blender selection methods work in the Ipo Curve editor. RMB click on a curve to select it. You can also use LMB on the color button next to its name. You can select many curves by using Shift RMB on every curve that is shown or by using Shift LMB on every color button. A curve that is selected shows its control points in white and its respective color button appear outlined in black (if it is the last selected when selecting multi curves the other button appear as pushed).

Options

RMB
: Select an Ipo curve under the mouse pointer

Select/Deselect All A
: Select all visible Ipo curves

Border Select B
: Select Ipo curves within an interactively mouse−drawn box

Curve Manipulation

Mode: All Modes

Menu: Ipo Curve Editor Curve Transform

Ipo curves can also be manipulated with many of the standard 2D transformations. The same axis locking options apply here too. Moving curves left and right will move them backwards and forwards in time.

Options

Grab/Move G
: Move the selected Ipo curve

Scale S
: Scale the selected Ipo curve

Curve Interpolation

Mode: All Modes

Hotkey: T

Menu: Ipo Curve Editor Curve Interpolation Mode

The interpolation mode determines how the Ipo values are calculated in between each curve point.

Options

Constant
: After each point of the curve, the value remains constant (horizontal). No interpolation takes place.

Linear
: Each curve point is connected by a straight line

Bezier
: The standard smooth curve interpolation

Curve Extend Mode

Mode: All Modes

Hotkey: E

Menu: Ipo Curve Editor Curve Extend Mode

The curve extend mode determines how the Ipo values are calculated outside the horizontal limits of the Ipo curve. This can be used to repeat a small section of animation or make animation continue endlessly.

Options

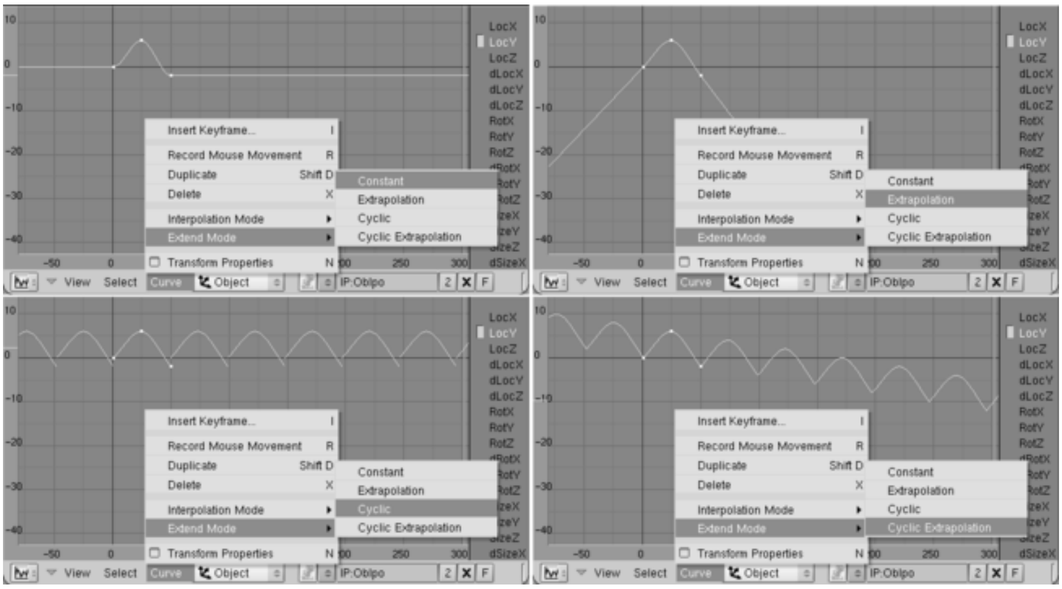
Constant
: The ends of selected curves are continuously (horizontally) extrapolated.

Extrapolation
: The ends of the selected curves continue in the direction in which they ended

Cyclic
: The complete width of the curve is repeated cyclically

Cyclic Extrapolation
: The complete width of the curve is extrapolated cyclically

Examples



Extended IPOs

Editing Curve Points

Mode: All Modes

Hotkey: Tab

Menu: Ipo Curve Editor Curve Edit Selected

As well as manipulating the curves themselves as a whole, you can also edit the individual points that define the curve. Curve points are added when a key frame is inserted (usually with the Insert Key I menu), and these points can be manipulated in time (X axis) and value (Y axis). Hit Tab to go in edit mode and be able to select each curve point independently. If you select one or more points with RMB, you can then move (G), scale (S) or rotate (R) them. You can also select and edit handle points that allow to finer control the interpolation curve. you can also set points values numerically by hitting N then entering a value in the "Vertex X" and "Vertex Y" fields inside the Transform properties panel.

Adding Curve Points

Mode: All Modes

Hotkey: Ctrl LMB / Shift D

Menu: Ipo Curve Editor Point Duplicate

Alternatively to adding curve points via inserting keyframes (eg. in the 3D View or buttons window), curve points can also be added manually in the Ipo Curve Editor by using Ctrl LMB. The point will be added to the active curve (the one with its color button next to its channel name outlined in black even one that doesn't

have any curve point already set), at the location of the mouse pointer.

Points are added according to the following rules:

- There is no Ipo datablock yet (in this window) *and* one *channel* is selected: a new Ipo datablock is created along with the first Ipo Curve with one vertex placed where the mouse was clicked.
- There is already an Ipo datablock, and a *channel* is selected without an Ipo Curve: a new Ipo Curve with one vertex is added.
- There is already an Ipo datablock, and a *channel* is selected with an existing Ipo Curve: A new point is added to the selected Ipo Curve linked with the nearest points before and after it. If multiple curves are selected, and/or you are in Edit mode, the points are added to the *active* curve (the one with the button next to its channel's name outlined with black).

Selected curve points can also be duplicated. Note that this will not duplicate the curve segments, but the points themselves, connected to the original curve.

Hints

Repeated Ctrl LMB clicking along a curve can be a good way to add randomness and unpredictability to an animation

Point Handle Types

Mode: All Modes

Hotkey: H, Shift H, Alt H, V

Menu: Ipo Curve Editor Point Handle Type

Similarly to 3D Curve objects, Ipo curve points possess two handle points by each side of it. Those handle points can have a variety of types, that can be used to specifically control the interpolation of segments between points.

Options

Free Handle H (black)
: The point handles are unlinked and can be manipulated in any way. (H toggles between Free and Aligned)

Aligned Handle H
: The point handles are linked in a straight line. (H toggles between Free and Aligned)

Vector V
: Both ends of a handle always point to the previous or next handle

Auto Shift H (yellow)
: This handle has a completely automatic length and direction, based on the points' proximity to each other

As soon as handles are rotated, by moving one of the point's handles, the type is changed automatically:

- Auto Handle becomes Aligned.
- Vector Handle becomes Free.

Ipo Keys

Mode: All Modes

Hotkey: K

Menu: Ipo Curve Editor View Show Keys

As easy as it is to work with animation curves, some people find it even easier to manipulate the keys. Changing the Ipo Editor to keys display makes two things happen:

- The Ipo Curve Editor now draws vertical lines through all the points of all the visible curves (curves are now shown in black). Points with the same 'frame' value are linked through the vertical lines. The vertical lines (the "Ipo Keys") can be selected, moved or duplicated, just like the points. You can translate the keys only horizontally.
- The object is not only shown in the 3D View in its current position but 'ghost' objects are also shown at all the key positions. On some video displays, you may have to press K in the 3D View window. In addition to now being able to visualize the key positions of the object, you can also modify them *in* the 3D View. For example, you can move the selected Ipo Keys.

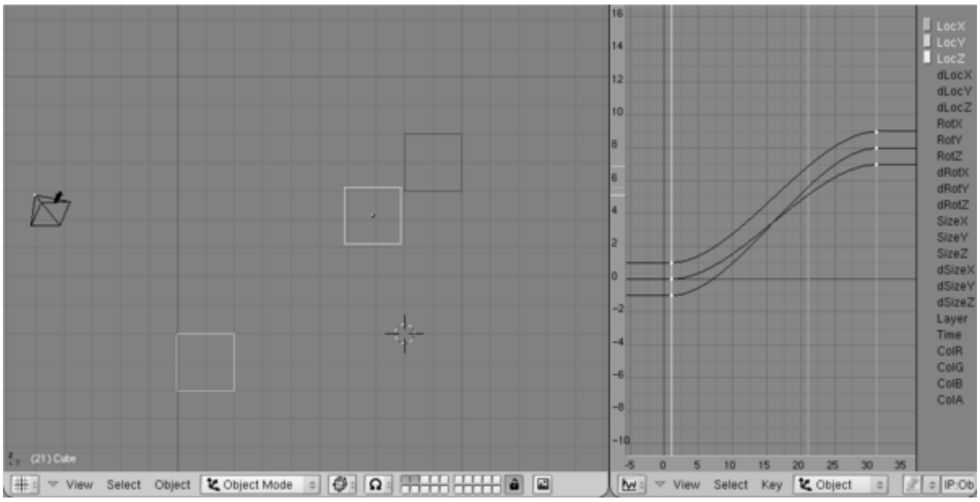
Onion Skinning and Ghosting

It is possible to see the keyed positions of an object in the 3D View. With the IPO editor in Keyframe mode, press K in the 3D View window. The location of the object will be drawn as ghostly outlines. The ghost location of the selected key will be drawn in a highlight color. Using this feature, you can physcially see the location, or path, that your object will take in 3D space.

Hints

- Use keyframe mode to change where an object is in time. For example if you currently have an object in a certain position in frame 100, but find that it gets there too early, go into key mode, select that key at frame 100, and Ggrab it and move it to the right. Click to drop it into position.
- Holding Ctrl down while dragging keyframes makes them snap to even frames.
- The I "Insert Key" always affects *all* selected objects. The IPOKeys for multiple objects can also be transformed simultaneously in the 3D View. Use the Shift K command: Timeline â†’ Show and Select Keyframes to transform complete animations of a group of objects all at once.
- Use Page Up â‡ and Page Down â‡Ÿ commands to select subsequent keys in the 3D View.
- You can create Ipo Keys with each arrangement of channels. By consciously *excluding* certain channels, you can force a situation in which changes to key positions in the 3DWindow can only be made to the values specified by the visible channels. For example, with only the channel LocX selected, the keys can only be moved in the X direction.
- Each Ipo Key consists of the vertices that have *exactly* the same frame value. If vertices are moved manually, this can result in large numbers of keys, each having only one curve. In this case, use the J ("Join") command to combine selected IPOKeys. It is also possible to assign selected IPOKeys vertices for *all* the visible curves: use I in the IPOWindow and choose "Selected keys".
- The DrawKey option and the IPOKey mode can be switched on and off independently. Use the button EditButtons−>DrawKey to switch off this option for this object. You can switch IPOKey mode on and off yourself with K in the IPOWindow. Only K in the 3DWindow turns on/off both the DrawKey and IPOKey mode.

Examples



The IPOKey mode

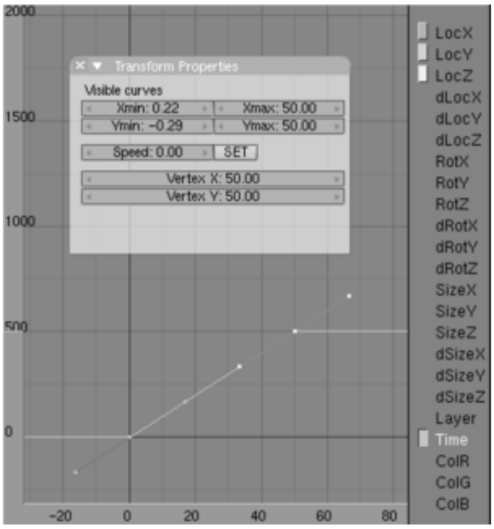
Special Notes on the Time Curve

Mode: All Modes

With the Time curve you can manipulate the animation time of objects without changing the animation or the other Ipos. In fact, it changes the mapping of animation time to global animation time (*Linear time IPO*). The Time curve is a channel in the Object Ipo.

In frames where the slope of the Time curve is positive, your object will advance in its animation. The speed depends on the value of the slope. A slope bigger than 1 will animate faster than the base animation. A slope smaller than 1 will animate slower. A slope of 1 means no change in the animation, negative power slopes allow you to reverse the animation. The Time curve is especially interesting for particle systems, allowing you to "freeze" the particles or to animate particles absorbed by an object instead of emitted. Other possibilities are to make a time lapse or slow motion animation.

Examples



Linear time IPO

To grasp this concept, make a simple keyframe−animation of a moving object, from a position to another in, say, 50 frames. Then select the Time channel and create a TimeIpo in the IpoWindow going from point (1,1) to point (50,50). It is easy to set the start and end point of an IPO by using N and entering the values numerically.

Multiple Time IPOs

You need to copy the TimeIpo for every animation system to get a full slow motion. But by stopping only some animations, and continue to animate, for example, the camera you can achieve some very nice effects (like those used in the movie "The Matrix")

Previous: Manual/Editing Ipos	Contents	*Next:* Manual/The Timeline

User Manual: Contents | Guidelines | Blender Version **2.43**

Blender really excels at animation, and the Timeline window is like having your own VCR controls at your fingertips. The timeline window, identified by a clock icon, is by default in the SR:4−Sequence screen layout just above the button window.

Timeline Window

The X−axis of the window is the timeline of your animation in frames. Frames are the fundamental unit of measure of time in Blender. LMB −clicking anywhere in the window brings Blender and all the other windows to that frame in time. Moving the mouse while holding MMB or Alt LMB pans the timeline to the left or right. Holding Ctrl MMB scales the display (frame range) in or out by moving the mouse left or right, just like in the 3D window.

Any Ipos (keyframes) for the currently selected object appear as vertical yellow lines at the frame they occur.

The exact frame is shown in the frame counter in the Timeline header, as well as the Buttons header. To the right of the frame counter are your standard VCR controls to rewind, skip back, play or pause, skip forward and skip to the end, and record. The start and end are set using the fields to the left of the frame counter, and synchronize automatically with the start and end frames in the Render buttons underneath the Anim button.

The menu consists of choices to View, Frame or Playback

View Menu

This menu controls what you view and how you view it:

- maximize display is a standard window feature to make this window full−screen.
- If Lock Time is enabled, all other windows are automatically synchronized to this window as the master clock.
- View All automatically zooms the display to show all events (Ipo keyframes) in range, so you don't have to pan.
- Center view centers the display on the current frame.
- Jump command do the same as the buttons in the header.
- Based on the frames per second you have set in the Render buttons, toggling the Show Seconds changes the display from frame units to seconds and fractions thereof. Very often storyboards are laid out in seconds; choosing this display unit makes things a little easier than doing all that multiplying in your head.
- Play back plays the animation from start to end. It keeps playing until you click the pause button (the two vertical lines).

Frame Menu

The Frame menu does something using the frame of animation:

- Set as... sets the current frame selected as the start or end frame as your range.
- Setting a Marker creates an orange pointer. Controlling the name of the marker by Ctrl M allows you to give a meaningful name to what is happening at that moment in time, such as "Boy loses Girl" or "Soda can spritzes". Using these Markers allows you to annotate the storyboard and animation sections. Of course, you can Grab, delete and duplicate markers.

Playback Menu

Playback controls how the animation is played back, and where.

- Set Frames/Sec allows you to change the speed of your animation.
- Windows enable the types of windows to be updated with each frame during playback. Otherwise, to conserve CPU power and allow smooth playback, some window updates can be disabled.

Markers

Markers: small but useful

Markers are used to denote frames at which something significant happens − it could be that a character's animation starts, the camera changes position, or a door opens, for example. Markers can be given names to make them more meaningful at a quick glance. When you create a marker in the timeline window, it also appears in the IPO, Action, NLA, and Audio windows, and vice versa, allowing you to see where the significant points of your animation lie, whichever window you are in.

Creating Markers

Mode: All

Hotkey: M

Menu: Timeline â†’ Frame â†’ Add Marker

The simplest way to add a marker is to move to the frame where you would like the marker to appear and press M.

Alternatively, you can either press Alt A or the Play Timeline button to make the animation play, and then hit M} at the appropriate points. Once you stop the playback, the markers will appear. This can be especially useful for marking the beats in some music.

Once you have made a marker, use RMB to select it, and Ctrl RMB to select multiple markers.

Naming Markers

Mode: All

Hotkey: Ctrl M

Menu: Timeline â†’ Frame â†’ Name Marker

Having dozens of markers scattered throughout your timeline won't help you much unless you know what they stand for. You can name a marker by selecting it, pressing Ctrl M, typing the name, and pressing the

Okay button.

Moving Markers

Mode: All

Hotkey: G

Menu: Timeline â†’ Frame â†’ Grab/Move Marker

Once you have one or more markers selected, press G to move them, and confirm the move with LMB or ENTER. If you hold Ctrl while moving them, the markers will move in steps corresponding to one second − so if you have 25 fps set, the markers will move in 25−frame steps.

Deleting Markers

Mode: All

Hotkey: X

Menu: Timeline â†’ Frame â†’ Delete Marker

To delete the selected marker(s) simply press X and confirm with LMB .

Previous: Manual/Ipo Curves and Keyframes	Contents	*Next:* Manual/Following a Path

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.43**

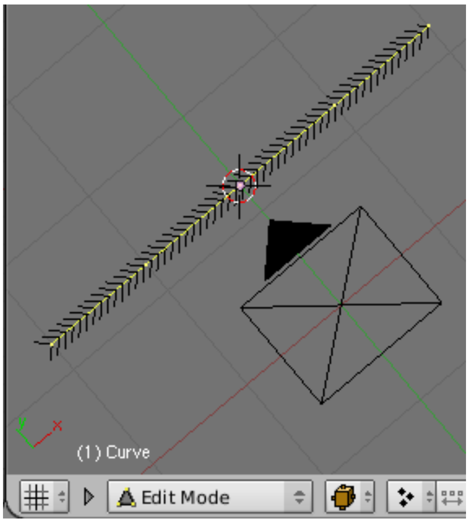
A different way to have Objects move in the space is to constrain them to follow a given path. Sometimes objects need to follow a specific path through a minefield, or it is just too hard to animate a special kind of movement with the keyframe method. For example, a planet following its way around the Sun − animating that with keyframes is virtually impossible. In these cases, a special type of curve object called a *Path* can be used as the path for another object to follow.
There are two steps in animating an object to follow a path:

1. Create the Path
2. Tell the object to follow that Path

Creating a Path

Mode: 3D View Object Mode

Hotkey: space Add Curve Path



Path in Edit Mode

A Path is a special type of Curve object. Blender has a specific Path curve object, added as indicated in the reference box above, but any curve can be used as a path. So, for example, if you used a curve to show the groove in the toy track of a electric race car, you can use that same curve as the path for the race car itself. The two act the same, however there are some automatic features that make using a Path easier than a generic Curve.

The Path is made up of control points, initially 5, and has a linear direction shown by the the arrows in Edit mode. You can close a path by pressing C to make the path cyclic. In the example to the right, the path is going in the +X direction in a straight line (left to right).

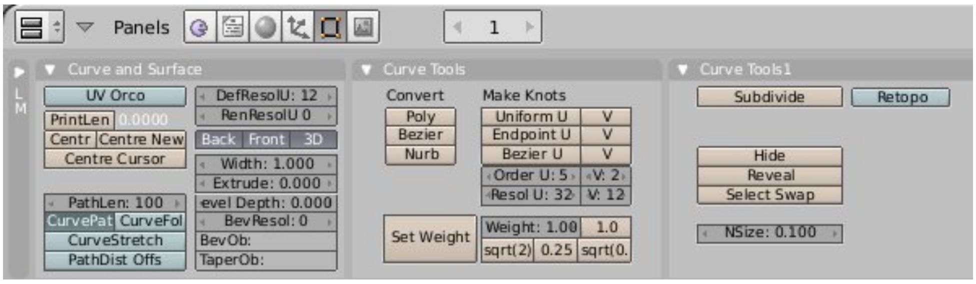
Speed IPO: When you add a Path, a Speed IPO curve is also automatically added, shaped such that any object traversing the path smoothly accelerates at the start and then decelerates smoothly to a stop at the end of the path over 100 frames. You can view this Speed IPO curve in the IPO window, Path IPO type. This Speed curve overrides the Pathlen value. You can edit this IPO curve to change the speed and location of the object over whatever time period you wish, using the standard IPO editing features of Blender. The Y value represents the "percent complete" of the object along the path; 0.0 is the start of the path, and a value of 1.0 is the end. The X value is the frame of the animation. Thus, a sine−wave−shaped (sinusoidal, for you geometry

geeks) Speed curve will make any objects that follow the path to move back and forth along the path in a very hypnotic fashion.

Like all objects, Paths are named in the Links and Materials panel or in the Properties panel, by default "Curve". This becomes important when you constrain an object to the path, because you have to enter the path's name in the Ob: field in the constraining panel. The older way to make an object follow a generic curve is to parent the object to the curve. This legacy technique is described below. The rest of this paragraph discusses using the Path object and Object Constraint method.

Select the object which you want to follow the path. In the Object (F7) context Object buttons, click Add Constraint−>Follow Path. In the Target OB: field, enter the name of the path. The Constraint name button should turn from red to grey, and the center of the object will jump to the start of the path. Click here for more information on the constraint (see also its dedicated page in this manual).

Path Buttons Panels



The Curve/Path Editing Panels

The path has all of the normal curve tools available. Focusing on useful Path features, working left to right through the panels:

Curve and Surface Panel
: **CurvePath** designates the curve as a path that can be followed (on by default). Objects can be translated (location) by the path, and if **CurveFollow** is enabled, also rotate to point along the path, like a train on a track. Similar to a Lattice, **CurveStretch** makes the object stretch, like a rubber band, along the path. **Pathlen** defines the default number of frames over which the object will traverse the path. A Speed curve provides much more control and, if present, overrides the Pathlen value.

Curve Tools Panel
: You can change the type of curve (Poly, Bezier, Nurbs) which changes the way the control points influence the path. Each Control point has a weight, or influence, on the speed of the object along that part of the path. The preset buttons save you from typing numbers in the numeric field. The square root buttons calculate smooth acceleration/deceleration using the square root of 2 as a speed factor between points. Be sure to click Set Weight to lock in your change to that point.

Curve Tools 1 Panel
: The Subdivide button is useful for inserting more control points between selected control points.

Changing the Path

Mode: 3D View Edit Mode

Hotkey: G to Grab, E to Extrude, C to toggle Cyclic

Menu: CurveExtrude; CurveToggle Cyclic, CurveSegmentsSwitch Direction

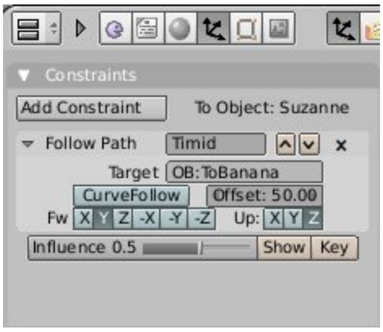
All the normal vertice editing controls are available, mostly Grab. Special Curve tools (like Cyclic, which make a closed path) are also available. Of special note:

Extrude
: used to extend either the start or the end of the path. With the start or end point selected, either Extrude or Ctrl LMB click to add points to that end of the path.

Switch Direction
: swaps the path start and end points

Anim settings Panel
: the TimeOffset numeric at the bottom of this panel specifies when in time this path starts influencing its objects

Following a Path



Path Constraint on Object

With your object selected, add a Follow Path constraint to the object. Enter the name of the Path, and specify the orientation. Paths have a local XYZ orientation, and the buttons align your object's local orientation to the path. In this example to the right, our favorite monkey has a "Timid" constraint to follow the path named "ToBanana". The path only influences her motion by about 50%, so she does not follow the whole path, but is only displaced by half the path length. She starts along the path to the banana at frame offset 50. The actual start frame is this offset plus the Path's animation setting for TimeOffset.

When you add a constraint, the object moves relative to the path, but does not have to physically follow that path in the same 3D space. It can be physically apart (beside, away from) the actual curve. This allows you to have a generic path, like "Falling in the wind" and then model many Leaves ("Leaf.001", "Leaf.002", etc.). Each leaf is moved away from the path and thus follows the path, but in its own space. Any Ipo Curves for that object are still in effect as it moves along the path, allowing you to let the object swivel left or right as they walk along the path, like a cartoon bird spinning and sliding on the ice.

Speed/Progress via Time Curve

You control the amount of time (actually the object's percentage progress) by RMB selecting the curve and, in the IPO window, LMB selecting the Time channel. Then Ctrl LMB click to add interpolation points at various frames, with the Y value between 0 and 100 reflecting the progress or distance along the path the object should be at that time (frame). If the curve is increasing, motion is forward; if the curve is decreasing, motion is backward. Thus, a simple sine wave (extended cyclic) curve will make an object swing back and forth along the path forever.

Paths can follow other Paths:

Just add the FollowPath constraint to a path. For example, a standard "out of the ballpark home run" baseball trajectory can be offset by the wind path.

Legacy Path Method

V.: 2.37−

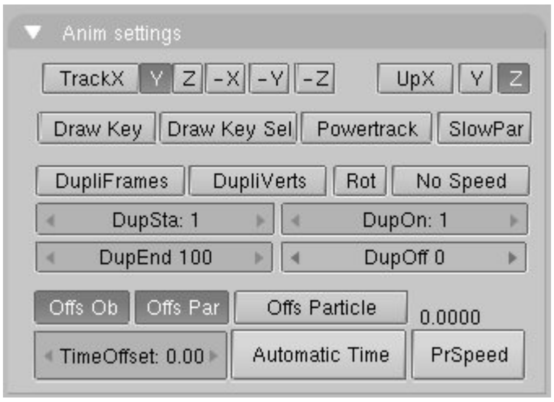
Editor's Note

The following text was in place prior to Version 2.43. It is preserved here for users of older versions, and for working with old blend files that use the old method of path animation. The spaceship example is still relevant and helpful in explaining/providing more detail.

As for tracking, there are two Path animation methods, the old, pre 2.30 method described here and the new method, which actually defines a constraint, which will be described in *character_constraints*.

When parenting an Object to a Curve, you will be asked to choose a Normal Parent or a Follow Path option. The Former is what you need for conventional Path animation, but other actions needs to be taken later. The second option creates a "Follow Path" Constraint, and it is all you need to do. If, when following a path, the dashed line between the object and path goes to the wrong end of the path simply clear the parent, select the path and go into edit mode. Select all the control points and select, in the Curve Menu, segments and select switch direction, then re−parent the object to the path. Any kind of curve can become a path by setting the option CurvePath Toggle Button in the Animation Buttons window (F7) to ON (*The Action Window with Path Buttons.*).

When a Curve has children objects it can be turned to a Path by selecting it, going int0 the Editing Context (F9) and activating the CurvePath Toggle button in the Curve and Surface Panel. Child objects of the Curve will now move along the specified path. It is a good idea to set the Curve to 3D via the 3D Toggle Button of the Curve Edit Buttons so that the paths can be freely modeled. Otherwise, in the ADD menu under Curve−>Path, there is a primitive with the correct settings already there. This is a 5th order NURBS spline, which can be used to create very fluid, continuous movements. Normally a Path is 100 frames long and it is followed in 100 frames by children. You can make it longer or shorter by varying the PathLength: Num Button. The *speed* along a path is determined with an appropriate curve in the IPO Window. To see it, in the IPO Window Header button you must select the Curve type for the IPO block. A single channel, Speed is there. The complete path runs in the IPO Window between the vertical values 0.0 and 1.0. Drawing a curve between these values links the time to the position on the path. Backward and pulsing movements are possible with this. For most paths, an IPO Curve must run *exactly* between the Y−values 0.0 and 1.0. To achieve this, use the Number menu (N) in the IPO Window. If the IPO Curve is deleted, the value of PathLen determines the duration of the path. A linear movement is defined in this case. The Speed IPO is a finer way of controlling Path length. The path is long 1 for time IPO, and if the time IPO goes from 0 to 1 in 200 frames then the path is 200 frames long. Using the option CurveFollow, in the Curve and Surface Panel, a rotation is also given to the Child objects of the path, so that they permanently point in the direction of the path.



Tracking Buttons

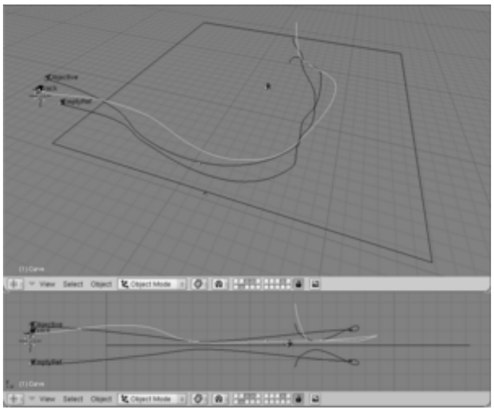
Use the "tracking" buttons in the Anim settings Panel of the Object (F7) context to specify the effect of the rotation (*Tracking Buttons*) as you would do for Tracking:

- TrackX, Y, Z, −X, −Y, −Z This specifies the direction axis, i.e. the axis that is placed on the path.
- UpX, UpY, UpZ Specifies which axis must point 'upwards', in the direction of the (local) positive Z axis. If the Track and

the Up axis coincides, it is deactivated.

Note

Curve paths have the same problem of Beveled curves for what concern the definition of the "Up" direction.

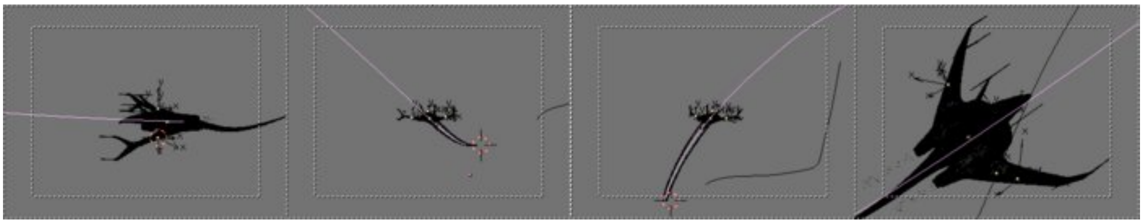


Complex path animation (Click for larger image)

To visualize these rotations precisely, we must make it possible for a Child to have its own rotations. Erase the Child's rotation with Alt R. Also erase the "Parent Inverse": Alt P. The best method is to 'parent' an *unrotated* Child to the path with the command Shift Ctrl P: "Make parent without inverse". Now the Child jumps directly to the path and the Child points in the right direction. 3D paths also get an extra value for each vertex: the 'tilt'. This can be used to specify an axis rotation. Use T in EditMode to change the tilt of selected vertices in EditMode, e.g. to have a Child move around as if it were on a roller coaster. *Complex path animation* shows a complex application. We want to make a fighter dive into a canyon, fly next to the water and then rise again, all this by following it with our camera and, possibly, having reflection in the water! To do this we will need three paths. Path 1 has a fighter parented to it, the fighter will fly following it.

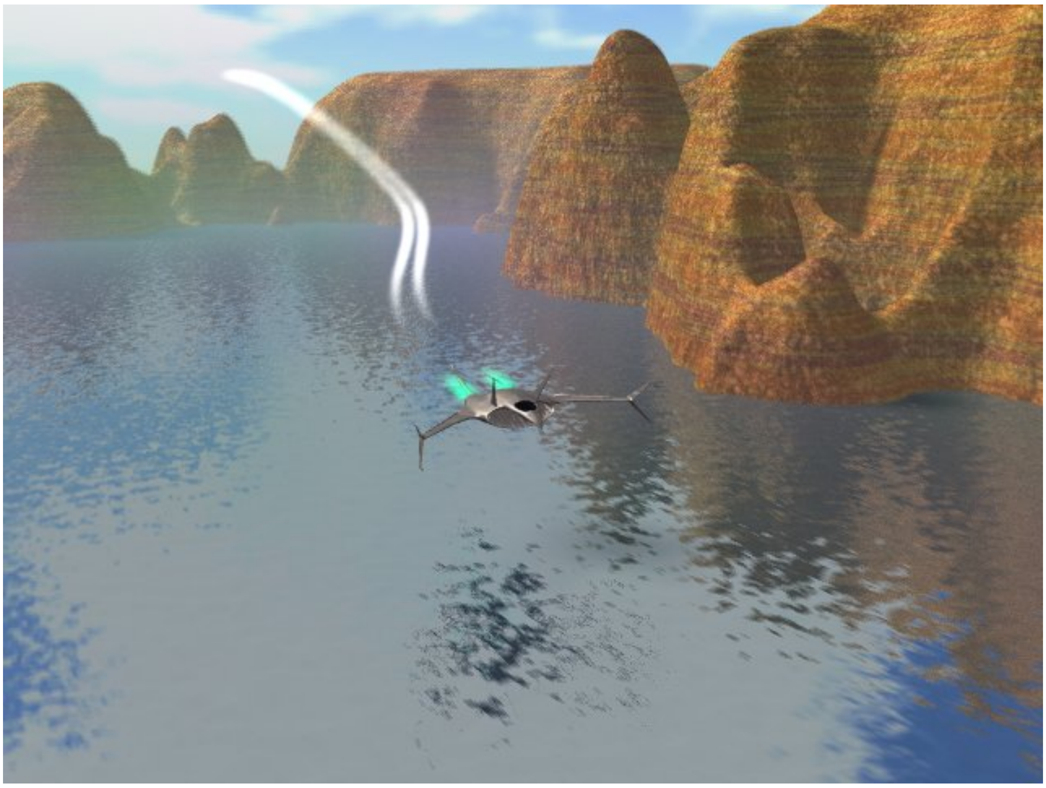
 Legacy Path Method

The fighter has an Empty named 'Track' Parented to it in a strategic position. A camera is then parented to another curve, Path 2, and follows it, tracking the 'Track' Empty. The Fighter has a constant Speed IPO, the camera has not. It goes faster, then slower, always tracking the Empty, and hence the fighter, so we will have very fluid movements of the camera from Fighter side, to Fighter front, other side, back, etc. (*Some frames, the camera fluidly tracking the fighter*.)



Some frames, the camera fluidly tracking the fighter. (Click for larger image)

Since we want our fighter to fly over a river, we need to set up an Env Map for the water surface to obtain reflections. But the Empty used for the calculations must be always in specular position with respect to the camera... and the camera is moving along a path! Path 3 is hence created by mirroring path 2 with respect to the water plane, by duplicating it, and using M in Edit Mode with respect to the cursor, once the cursor is on the plane. The Empty for the Env Map calculation is then parented to this new path, and the Time IPO of Path 2 is copied to Path 3. *A frame of the final animation.* shows a rendered frame. Some particle systems were used for trails. The scene presents many subtle tricks, as particles for the jet streams, fog, a sky sphere encircling the scene and so on.

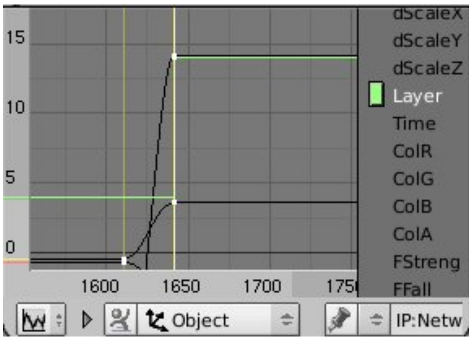


A frame of the final animation.

User Manual: Contents | Guidelines | Blender Version **2.43**

Many times you want an object or actor to wait off stage until it is their time to come on and do their thang. In Blender, you commonly render only a few of the 20 available layers. Further, you can set some lamps to illuminate only objects on their layer. By changing an object's layer during the animation, it is possible to animate an object's layer "membership" and thus it's visibility in the scene, and its lighting.

Further, by putting the object on a layer that is not selected in your 3D view, it will not clutter up your workspace. During the frames of the animation when it is on a visible layer, you will see it an be able to work with it.



Changing from Layer 4 to 14

To animate an object's layer membership, go to the frame when you want the object to appear. Ensure the object is on a visible layer (Move Layer command will tell you what layer it is on). Insert an Ipo Key, and choose Layer (the menu option located below the Location and Rotation selections). In your Ipo Window, you will see the Layer channel now has a value corresponding to the Layer. Back up one frame, Move the object to an unselected (invisible) layer. Now, here's a slight glitch; you cannot add a Layer Key because the object is on an unselected layer. So, temporarily enable that layer by shift−selecting it from the 20 Layer buttons, Insert another Layer key, and then de−select the undesired layer.

Select All Layers

To activate all Layers, press ` (accent grave .. top left of US keyboards)

Scrub or play through the animation and you should see your object appearing at the indicated frame. You can edit this Ipo control point, usually in Keyframe mode, to change when the shift occurs.

Tips

You will note that the Layer Ipo "curve" is a discontinuous jump. There is no way to have an object gradually change layers. To make an object fade into view, you have to animate its Material Alpha setting to fade in from 0.0 to 1.0.

There are two ways to check which layer an object is on :

− Press M to display the layers menu, the object's layer(s) buttons will be enabled. Press ESC or OK to dismiss the menu

− Display the Buttons Window / Object Buttons F7. Under the Draw panel, the Layers buttons displays the object's active layer(s). The active layer(s) can be changed from that panel as well, instead of using the

floating layer panel M

Previous: Manual/Following a Path	Contents	*Next:* Manual/Animation Deformations

User Manual: Contents | Guidelines | Blender Version **2.43**

Animating an Object's location, rotation, scale are trivial animations. Animating its Material settings is a bit more conceptually challenging. So let's take it to the next level, shall we? With Blender, you can change, reshape, and deform your objects over time!

There are many ways of achieving this actually, with increasing levels of complexity. However, each level builds on one another and as usual, there are many ways to do it. Depending on how fancy you want to get and what you want to do, one approach may be preferable over another.

- Hooks use an Empty (or other object, but most commonly an empty) to pull at selected vertices of your mesh or curve or lattice or NURBS surface. You animate the location of the Hook to animate the deformation of your mesh. Hooks are modifiers and are discussed in the Modifer section of this manual.
- Relative Shape Keys change the location of some of the vertices of a mesh. Relative shape keys can be combined, sometimes called BlendShapes, because different shapes are blended together. For example, a face mesh can have one shape key to smile, and another to blink. Those two action can be combined so that it smiles and blinks at the same time.
- Absolute Shape Keys allow you to completely deform and change the mesh, even by adding or subtracting vertices over time.
- Lattice Animation uses a lattice as a cage that uniformly squeezes or expands a mesh, like the event horizon in a black hole, or as a comical cartoon car screeching to a stop. You define shape keys for the Lattice, which in turn affects the target mesh as it moves through the lattice.

The latter method, Lattice Animation, brings up an interesting feature in Blender, in that one object can deform another. This feature has application especially to Bevel and Taper objects. These are curves that affect other objects, like meshes or text objects. Changing the shape of these curves over time, using shape keys, will cause other objects (the mesh, text, etc) to change shape as well.

Previous: Manual/Object Layer Animation	Contents	*Next:* Manual/Shape Keys

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.43**

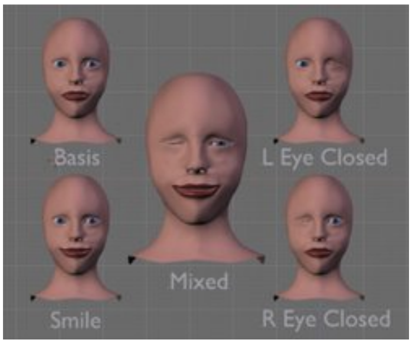
Shape keys store different shapes of the same mesh. In other 3d applications they are called 'morph targets' or 'blend shapes' or even 'vertex keys' in older versions of Blender.

Blender offers both **relative** shape keys and **absolute** shape keys. *Relative* shape keys, discussed on this page, record each shapes **difference** (a.k.a. "delta") relative to a base shape. *Absolute* shape keys record directly each different shape.

Mode: Object Mode

Panel: Editing Context Shapes

Description



An Example of Mixed Shape Keys

Relative Shape keys can be blended on a percent basis with other shape keys to achieve the desired effect. Take for example a face. The user may model a face with a neutral expression and have shape keys for a smile and a frown. These shape keys could then be combined in the Action Editor and your talking face could have some serious emotional problems.

Use shape keys for animating faces or when you want the shape of any mesh to be able to change over time; for example, a ball squishing, or a soda can being crushed. When animating faces, suggested shape keys are:

- Mouth forming each sound shape
- Lips forming a smile and frown
- Eyes: Blinking, showing basic emotions, and even winking.

Note that a shape key does *not* say how fast the shapes are transitioned or in what order; you can make these keys in any order or speed you wish. Later, you will add Ipo keys that say exactly when the mesh takes shape.

Workflow

The overall process for using shape keys in Blender is:

1. . Define your basis shape.
2. . Identify the list of primitive morphs that you might want
3. . Create Vertex Groups for sets of vertices to participate in each morph family

4. . Define your target shape(s), one for each morph
5. . Using the IPO curves, define when you want the target shape to influence the basis shape.

Target shapes can be created at any frame; they are time−independent. They exert their influence according to the shape IPO curve. Each target shape has a channel in the IPO window. When a target shape's IPO curve is at 0.0 y value, it does not deform the mesh. As its influence increases, it deforms the mesh at that frame. Since the IPO channel is a transition (a smooth curve or a linear line or whatever options you choose for the IPO curve) you can control how fast the mesh morphs to the new shape.

Multiple shapes can simultaneously morph the basis mesh, each of them exerting their influence through their IPO curve, since each of them have their own channel.

If you want to change the basis shape, you will have to create a new shape, and using a script, copy the shape keys over to the new shape, adjusting them as you desire.

A morph family is a group of shapes related to an area of the mesh. For example, take the face. You have the eyebrow area (possible shapes are Surprise, Furrow, eyebrow lift−left, and eyebrow lift right), the eyelids (Blink), the mouth (smile, frown), nose (flare, wiggle) and tongue (ay, th, i, stickout, tunnel). The jaw open action may be controlled by a bone. In this case you would a vertex group for the eyebrow, another for the tongue, etc.

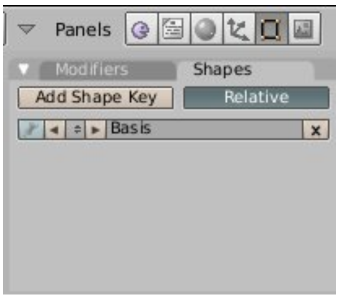
Creating Relative Shape Keys

Mode: Object Mode with Mesh, Curve or Lattice object selected

Panel: Editing Context Shapes

Hotkey: I

Menu: Object Insert Keyframe



Shape keys are all stored relative to a basis mesh. You can enable shape keys and insert the basis key by selecting the mesh object and pressing the I then choosing Mesh. Alternately, this step can be done with the Add Shape Key button in the Shapes panel. This panel is located as the second tab in the Editing F9 buttons in the Buttons window. In this panel, be sure that the Relative button is enabled.

The original shape key is shown in the Ipo Window, (Ipo Type: Shape), for relative keys as an ellipses just under the pushpin. The original absolute shape key is named "Basis". This difference alerts you to whether you are working with Relative or Absolute shape keys.

Once the basis shape key has been created, you can create additional shape keys in the same way. With the mesh selected, and in its deformed state, either press I and then choosing Mesh, or click the Add Shape Key

button in the Shapes panel. A new Shape Key will be created. If this is the first shape created after the basis shape, it will be called Key 1. Shift LMB into the name field in the panel to change the name to something meaningful.

An important thing to remember is that when creating a new shape, the new shape will be based off the current selected shape **WHEN YOU ENTER EDIT MODE**. When creating shapes from the basis shape, the new shape will be a copy of that shape, if the user creates a new shape while another altered shape is selected, the new shape will be a copy of that altered shape. This allows you to further edit selected shapes; simply leave edit mode to lock in that shape.

Fun Fact

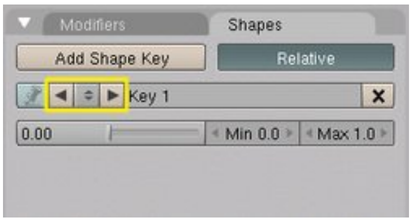
Old school Blender users sometimes refer to shape keys as Relative Vertex keys, or RVKs for short.

Editing Shape Keys

Mode: Object Mode

Panel: Editing Context Shapes

Various aspects of each shape key can be edited by activating it in the Shapes panel.



Location of the Shape Key selection tools in the Shapes panel

Arrows and the Menu

Choose the active shape key. The arrow buttons cycle through shapes, and the menu allows you to choose a specific shape directly. Note that when you first choose a key, it is visualized at full strength in the 3D View regardless of any key frames that have been set, so you can see exactly what shape you're activating. Changing the current frame will reset the 3D View to show the results of the shape key blending again.

Once the desired key is selected, the shape can be altered by entering editmode Tab and manipulating the vertices. **Note:** Adding or deleting a vertex once shape keys have been added is very tricky. The changes are propagated to the other shapes based on their position in the current shape, this can have drastic effects on other shapes.

Current Shape Key Name (text field)

Rename a shape key from its automatic name by Shift LMB in the name field and typing in a new name to change it. It's very important to keep your shapes organized and well named, so you can remember which one you're editing in the Action or Ipo editor. It's very easy to end up with a huge amount of shape keys when you start animating faces and you don't want to get your work confused.

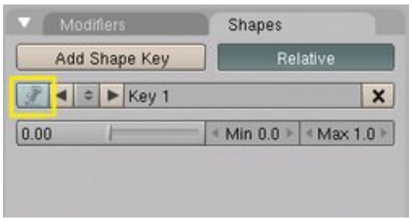
Propagating Changes to other Shapes

If you find an error in your basis shape *after* you have started defining other shapes, you can make your changes and, with those vertices selected, use the specials menu W −> Propagate to All Shapes" in Edit mode. Those vertex positions will replace the corresponding vertex positions in the other Shape keys.

Propagating Changes from another Shape

If you find an error in your basis or other shape *after* you have started defining other shapes, with those vertices selected, use the specials menu W −> Blend from Shape and choose which shape key to use as a reference for repositioning the vertex. You can then even choose to what extent to apply that reference point to the existing vertex location. Think of it as a way to average the position of vertices between shapes.

Pinning



Location of the Pin Icon in the Shapes panel

When a mesh object does not have a pinned key, it can show multiple keys at once. To prevent this, click on the pin icon.

The current mesh object is now locked at that shape key, and will only show that key. This feature is useful when multiple keys are affecting a mesh object and you would like to see the effect of one specific key.

One other use for pinning is to create a shape gallery.

Since Blender allows for the creation of Linked Duplicates which share underlying data (e.g. mesh and shape key data) multiple copies of a mesh can be created using Alt D.

Once a duplicate has been created, move the duplicate to a different area of the screen. Pin the duplicate to a shape key by clicking the pin icon in the Shapes panel. Finally, reselect the 'original' mesh. If this is done for each key, a shape gallery emerges (which also helps in editing keys).

Hints

Pinning Shape Keys can also help to speed up interaction with a mesh (especially when animating it with armatures). If a shape is pinned, Blender doesn't have to calculate the myriad interactions and blends of different keys in real time, which speeds up operation. This is particularly useful when using armature−driven shape keys to correct deformations, since it's usually not something critical that you need to see when animating. Remember to un−pin the shapes when you're done and ready to render though!

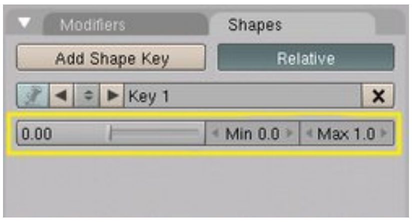
Blending Shape Keys

Mode: Object Mode

Panel: Editing Context Shapes

Shape Keys can be mixed in several different ways. These methods have the same net effect of creating IPO data, but are different in interface.

Blending Using the Shapes Panel



Shape Panel and Shape Keys

In the Shapes panel when a key is not pinned, it has an additional row of buttons; value, min and max.

Key Value (slider)
: Current value of that key at the current frame. Adjusting this slider will insert or adjust a control point in the shape key control curve at the current frame. If I put my "blink" shape key at 50% instead of 100%, I'll have half−open eyes.

Min and Max
: Adjusts the boundary values for the value slider both in the Shapes panel and the Action Editor Window. It is possible to use a maximum greater than 1.0, to push shapes to even greater extremes, but be careful, since you are exaggerating the shape key, and this can mesh up your mesh.

It is often easiest to to mix your shape keys in the Action Editor instead of doing it from the main editing panel.

Blending Using the Action Editor

Action Editor and Shape Keys

In the Action Editor, expand the Sliders button and use the sliders to adjust the key values at the current frame. The min / max values set in the Shapes panel are reflected in the Action Editor window sliders for that

shape.

If there is not already a keyframe at the current frame (indicated by the green line), a new keyframe will be added.

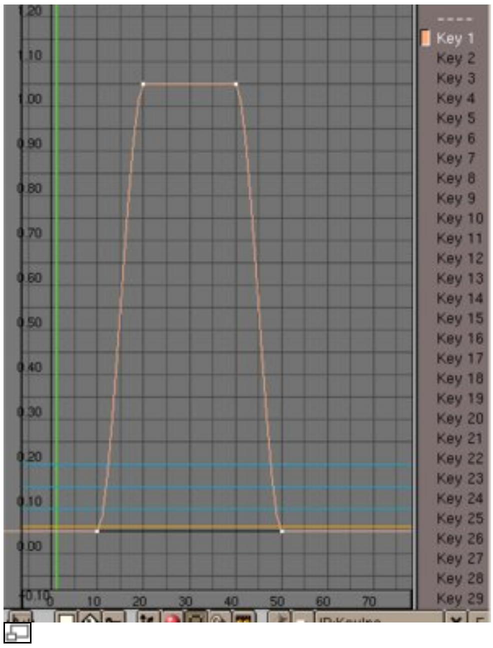
Blending Using the Ipo Curve Editor

The Ipo Curves show the underlying data that is being controlled by the other mixing methods. It is in this window where the values are mapped to frames for each shape key. In order to access the Shape key Ipos, select Shape from the Ipo type popup menu in the Ipo Curve Editor Window header. Each shape key will be shown in the Ipo list on the right side and can be altered in the same manner as any Ipo curve in Blender. See the "Editing Ipo Curves and Keyframes" section for more information.

The vertical order of the Vertex Keys (the blue lines) from bottom to top determines its corresponding Ipo Curve, i.e. the lowest blue key line will be controlled by the Key1 curve, the second lowest will be controlled by the Key2 curve, and so on. No Ipo is present for the reference mesh since that is the mesh which is used if all other Keys have an Ipo of value 0 at the given frame.

The Ipo Curve of Key 1 shows a mesh undeformed up to frame 10, then from frame 10 to frame 20 Key 1 will begin to affect the deformation. From frame 20 to frame 40 Key 1 will completely overcame the reference mesh (IPO value is 1). The effect will fade out from frame 40 to frame 50.

Keys in the IPO Window



The IPO curve of Key 1

Using IPO Drivers

See the Advanced Driven Shape Keys page for information regarding how to set up an IPO Driver.

Previous: Manual/Animation Deformations	Contents	*Next:* Manual/Absolute_Shape_Keys

User Manual: Contents | Guidelines | Blender Version **2.43**

Shape Keys (as opposed to Object keys, the position of the whole object center) are the specified positions of vertices *within* an Object; the actual points that define the mesh. Since this can involve thousands of vertices for an object, separate motion curves are not created for each vertex because it would overload your computer's memory. A more generic keyed position system is used instead; a single IPO Curve is used to determine how interpolation is performed and the times at which a Shape Key can be seen.

Mode: Object Mode / Edit Mode

Panel: Editing Context Shapes

Hotkey: I

Menu: Object Insert Keyframe

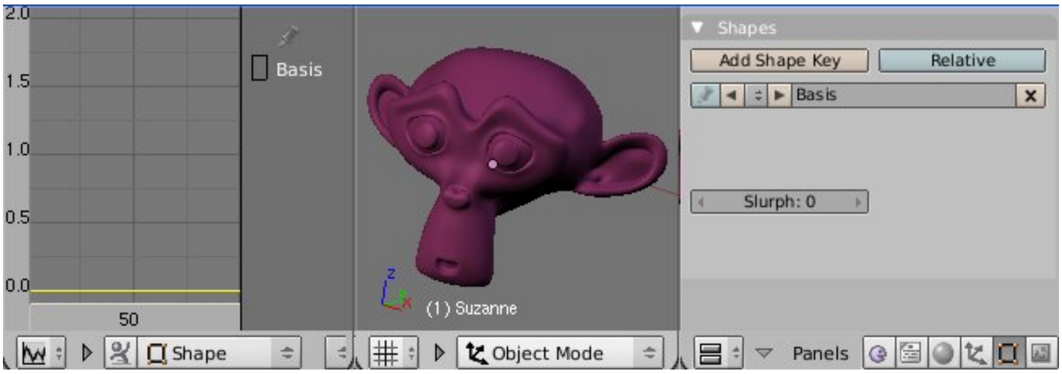
What Types of Objects can be Shaped?

You can define Absolute Shape Keys for:

- Polygonal Meshes,
- Bezier and Nurbs Curves,
- Nurbs Surfaces,
- and Lattices.

All of these objects have vertices. The interface and use for Shape Keys is the same for all of them. In general, you define a Basis key, edit the object to a new shape, leave edit mode and insert a key for each deformation. To animate the shape shift, you then say when (in terms of Frames) you want the object to assume that shape using the Action Editor.

Creating Absolute Shape Keys



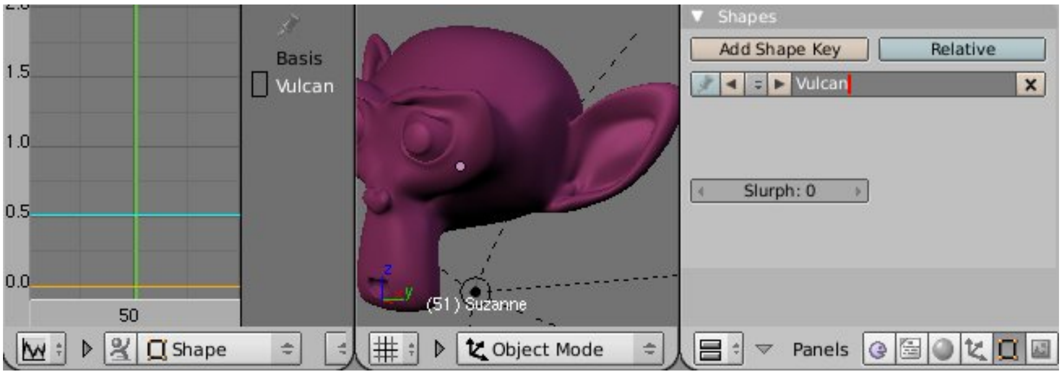
Basis Absolute Shape Key

The first absolute shape key that is created is always the *reference* or Basis key. This key is the Orange line in the Ipo Window (yellow if selected). Only if this Key is *active* can the faces and curves, or the *number* of vertices, be changed.

A few notes about the above picture: When working with Shape keys, it is very handy to have an IPO Window open. Use the first Animation Screen layout if you like (CTRL+1). In the IPO Window, we must

then specify that we want to see the Shape Keys by selecting that Ipo Type from the Ipo Window header. To do this, use the IPO type Menu Button and select Shape. Go to the 3DWindow and select the object (Suzanne, in this case) and press I. The "Insert Key" menu has several options, the last being Mesh. For other kinds of objects, this menu item will be named for that kind of object e.g. if a curve object is selected, the menu item on the Insert Key menu will be "Curve". The 3D View header menu is also context sensitive, and the menu name will change based on the type of object selected.

As soon as this has been selected, a new panel appears in the Editing panel (F9) in the Shapes panel. It has buttons to Add Shape Keys (what you just did), and a Relative button, enabled by default. To work with Absolute shape keys, disable this button. When it has been turned off (a lighter shade of green), Absolute Keys; an orange horizontal line is drawn in the Ipo Window in Shape display mode. This is the first key and thus the Basis key. The name of this shape key changes in the Ipo Window to "Basis".



Suzanne becomes Vulcan

To add succesive shapes, you

- Go to the frame in the animation where you want that shape to be fully formed
- Modify the shape (in Edit mode)
- Leave edit mode
- Click the Add Shape Key button (or do the I thing again in the 3D View).

For each key you add, blue−colored lines appear in the Ipo Window (cyan when selected). They are placed above one another in a height that is 1/100 of the frame number where they were created, just to space them out (a shape created at frame 50 will appear at 0.5).

ShapeKey creation

Creating Shape Keys in Blender is very simple, but the fact that the system is very sensitive in terms of its configuration can cause a number of 'invisible' things to happen. The following rule must therefore be taken into consideration. As soon as a Shape Key position is inserted it is *immediately active*. All subsequent changes in the Mesh are linked to *this* Key position. It is therefore important that the Key position be added *before* editing begins.

VertexKeys can be selected in the IPO Window or in the Shapes panel using the <> buttons. We always do this *out* of Edit Mode: the 'contents' of the VertexKey are now temporarily displayed in the Mesh. We can edit the specified Key by starting Editmode.

As you render your animation over the frame range, the shape changes from one to the other. You can verify this by scrubbing over the frame range using the Timeline window or by wearing out your arrow keys.

Sharing Shape Keys

Shape Keys are part of the Mesh data, not of the Object. When cloning an object, by making both objects use the same Mesh (Editing buttons, Links and Materials panel), the associated Shape Keys are also copied. Thus, it is possible to permit multiple Objects to share the same Shape Keys in Blender by making them use the same Mesh.

Workflow for Creating Absolute Shape Keys

Once the shapes are defined, there are three methods for working with Shape Keys:

The 'performance animation' method.

- This method works entirely in EditMode, chronologically from position to position:
- Insert Key. The reference is specified.
- A few frames further: Insert Key. Edit the Mesh for the second position.
- A few frames further: Insert Key. Edit the Mesh for the third position.
- Continue the above process...

The 'editing' method.

- We first insert all of the required Keys, unless we have already created the Keys using the method described above.
- Blender is *not* in EditMode.
- Select a Key. Now start EditMode, change the Mesh and leave EditMode.
- Select a Key. Start EditMode, change the Mesh and leave EditMode.
- Continue the above process....

The 'insert' method

- Whether or not there are already Keys and whether or not we are in EditMode does not matter in this method.
- Go to the frame in which the new Key must be inserted.
- Insert Key.
- Go to a new frame, Insert Key.
- Continue the above process...

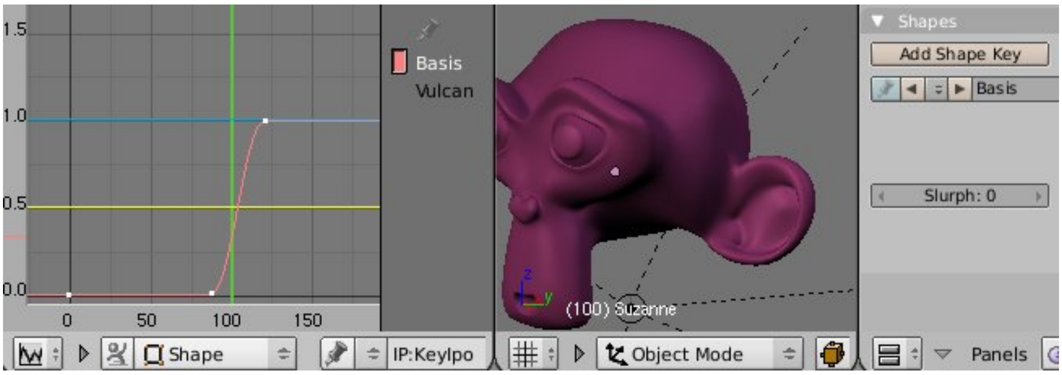
While in EditMode, the Keys cannot be switched. If you attempt to do so, a warning appears.

Shifting between Shapes

Now that you have your different shapes defined, you now want your mesh to shift between them at certain times in your animation. As you might have guess by now, there are a few ways to do it. I call them the line way, and the control way.

In the line way, in your Ipo Window, LMB click on a Shape Key; its name will turn white. Alternatively, scroll to the shape in the Shapes panel (its name won't turn white in the Ipo Window, but the box next to its name will shift to it, letting you know it is active). In the Ipo window workspace, Grab and move your mouse up or down, and drop the blue line at the frame you want, (on the y axis, remember: the y value corresponds to the 1/100 of the frame number where you want this key to have maximum effect).

In the control way, you define an Ipo curve which is a sort of Time (or Speed) curve for the mesh. This curve is linked to the "`Basis`" channel (or key) â€“ you can define a curve for any key, but only the basis one is useful (the others have no effect). This curve maps the current frame (x axis) to a 'virtual' frame (a position in the sequence defined by the absolute shape keys) on the y axis: to be clear, where the Ipo crosses a blue line, the corresponding key has its maximum effect; when the Ipo value is in between two keys (in between two lines), the mesh is deformed by the interpolation of this two keys (the closest the value to a line, the highest the influence of this key). So, for a linear curve with positive slope (a "climbing" one), the animation is the same as without any curve (it might just be faster/slower, and/or have a time offset). If you give your curve a negative slope, the animation will run backwards. If you make it go up−down, the animation will go forward and backward.



Using Control Points to Time the Shape Shift

In the example above, without any control, Suzanne would shape shift to a vulcan between frames 50 and 100 (yellow line at 0.5, blue line at 1.0). However, we selected the Basis key and added some control points (evidenced by the orange selector block). Thre are three control points, the important ones being at frame 80 with a value of 0, and at frame 120 with a value equal to the Vulcan key. The cursor (green line) is at frame 100. Normally, Suzanne would be a Vulcan by now, but because of the control points, she has only just started, and won't become a fully formed Vulcan until frame 120.

Options

- Linear: interpolation between the Keys is linear. The Key line is displayed as a dotted line.
- Cardinal: interpolation between the Keys is fluid, the standard setting.
- BSpline: interpolation between the Keys is extra fluid and includes four Keys in the interpolation calculation. The positions are no longer displayed precisely, however. The Key line is drawn as a dashed line.

Hints

- Key positions are *always* added with I in the 3D View or Materials buttons, even if they are located at the same position. Use this to copy positions when inserting. Two key lines at the same position can also be used to change the effect of the interpolation.
- If *no* Keys are selected, EditMode can be invoked as usual. However, when you leave EditMode, all changes are undone. So if you are frustrated because you edited your mesh and all your changes went away, remember you read this and you will remember why. Insert the Key *in* EditMode in this case to change the active key.
- For Keys, there is *no* difference between *selected* and *active*. It is therefore not possible to select multiple Keys.
- When working with Keys with differing numbers of vertices, the faces can become disordered. There

are no tools that can be used to specify *precise* sequence of vertices. This option is actually suitable only for Meshes that have only vertices such as Halos.
- If no IPO curves are defined on any of the absolute Shape Keys, the length of the animation defaults to 100 frames. That is, shape keys added past frame 100 will have no effect. An IPO curve must be defined on at least the Basis key for absolute shape key animations to exceed 100 frames.

Curve and Surface Keys

As mentioned earlier, Curve and Surface Keys work exactly the same way as Mesh Keys. For Curves, it is particularly interesting to place Curve Keys in the bevel object. Use the animation of that curve to, for example, animate a pumping artery, or a pulsing heart, or a growing worm, or balloon being inflated, or a tree growing. A scene in Elephant's Dream, where worms started growing (just before Proog conks Emo) used animated bevels and tapers while the curves extended.

Previous: Manual/Shape Keys	Contents	*Next:* Manual/Animation/Lattice Animation

User Manual: Contents | Guidelines | Blender Version **2.43**

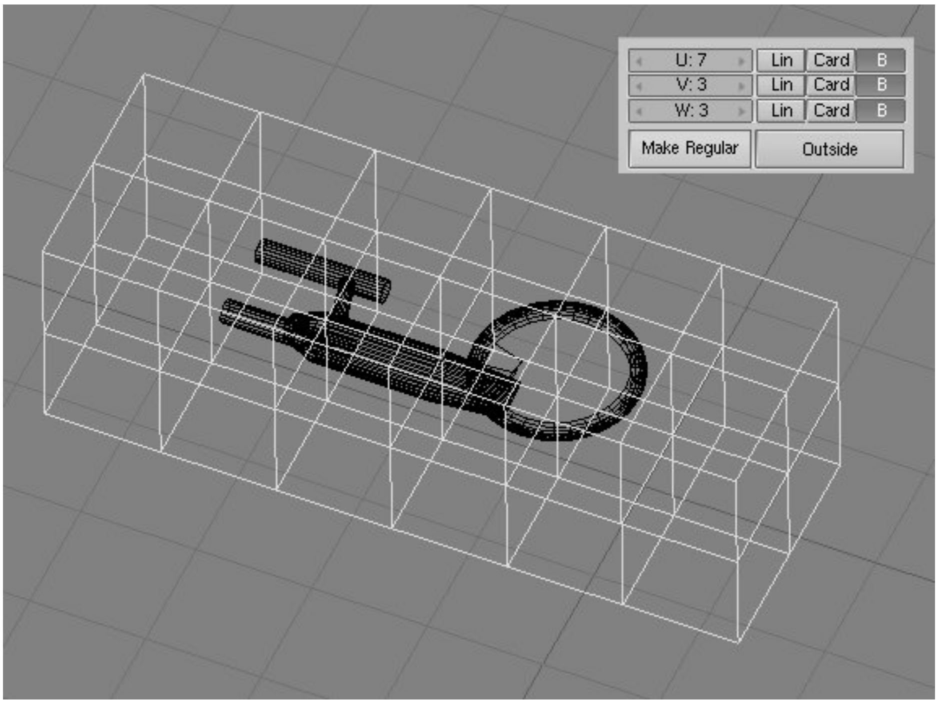
A lattice is a framework of controlling vertices. Associated a mesh to a lattice is a nice way to apply deformations to the mesh while modeling, but it is also a way to make animated deformations in time! To deform the mesh, you can deform the parent Lattice; the mesh deforms to fit inside the Lattice.

Lattice Animation

You can deform a Lattice in animations in a few ways:

- Animate the lattice vertices with shape keys
- Move the child object of the lattice through the lattice over time.

The first technique is basically nothing new than what contained in the previous Shape Key sections but applied to a lattice which has an object parented to it. With the second kind you can create animations that squish things between rollers, or achieve the effect of a well−known space ship accelerating to warp−speed. As the object enters the Lattice, it starts to deform; as it emerges, it expands like a foam rubber ducky back to its original shape.



Lattice setup.

For example, make a space ship mesh object. Add a lattice around the ship. Make the lattice with the parameters shown to the right in *Lattice setup*. This panel is shown in the Buttons window, Editing (F9) context.

There are two ways to make the lattice deform the spaceship:

- Select the object to be deformed by the lattice, and create a Lattice Modifier, entering the name of the Lattice in the Modifier panel.
- Make the object to be deformed a child of the Lattice through a special Parent relationship.

For the old school parenting approach, select the ship, then shift−select (holding Shift while selecting) the

Lattice, and press Ctrl P to make the lattice the parent of the ship. The popup window will ask if you want to make a regular parent relationship, or a Lattice Deform; choose Lattice Deformation.

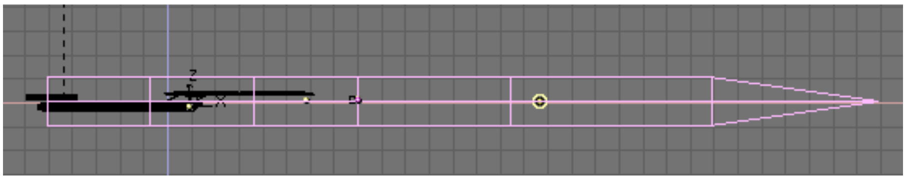
You should not see any deformation of the ship because the lattice is still regular. If you did, click the Regular button in the Lattice panel.

For the next few steps it is important to do them in EditMode. Select the lattice, enter EditMode, and select all vertices (A). Scale the lattice along its x−axis (press MMB while initiating the scale) to get the stretch you want. The ship's mesh shows immediately the deformation caused by the lattice (*Stretching*).

Stretching.

Now edit the lattice in EditMode so that the right vertices have an increasing distance from each other. This will increase the stretch as the ship goes into the lattice. The right ends vertices I have scaled down so that they are nearly at one point; this will cause the vanishing of the ship at the end (*Final lattice deformation*).

Select the ship again and move it through the lattice to get a preview of the animation. Now you can do a normal keyframe animation to let the ship fly through the lattice.



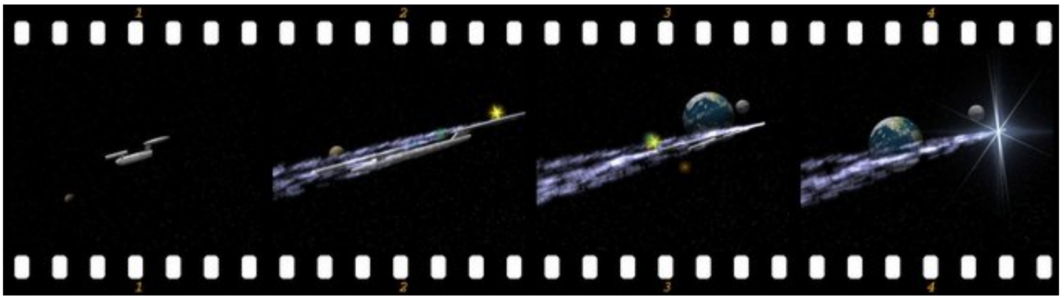
Final lattice deformation.

Implications for Camera Tracking

With this lattice animation, you can't use the pivot point of the object for tracking or parenting, since the vertices that make up the mesh are sucked into the Lattice or spit out, way off center from the object itself. When that happens, although the camera is looking at the mesh object's center, the mesh isn't there so the camera is looking in the wrong place. Instead, track an Empty which has an mesh object vertex as a parent.

To do this, create an Empty. Next, make sure the Empty is selected, then select the mesh object. Press Tab to enter EditMode. Select one vertex from the mesh object. Press Ctrl P to make the vertex the parent of the Empty. Press Tab to get out of EditMode.

Now tell the camera to track to the empty (via the Track To Constraint). Animate the location of the empty to lead the object as it enters the lattice, and lag as it emerges (or vice versa if your Lattice squashes the mesh). As the mesh enters the lattice and deforms, the empty will lead/lag its center, keeping the camera looking at the deformation action.



Some frames of the resulting animation.

Add some halos and a flash of light (an animated lamp) at the end, and you have something like that shown above.

Previous: Manual/Absolute_Shape_Keys | Contents | *Next:* Manual/Character Animation

Reference Guide: Contents | Guidelines | Blender Version **2.32**

FIXME

IPO Header

WindowType

As with every window header, the first button enables you to set the window type.

Menus

The triangular button expand/collapses menus. Menus provide a self−explanatory way to access all Blender functions which can be performed in the IPO Window. They are context sensitive and will change depending on the selected Object and current Mode.

Windows with menus, as the IPO Window, does not have the standard "Full Window" and "Home" header buttons, actions which have moved to the View Menu.

Menu items are in general self−explanatory and the relative functionality will be explained later on in the HotKeys section. The actions which do not have an hotkey are the Extrapolation settings :

Extend mode Constant

The ends of selected IPO Curves are horizontally extrapolated.

Extend mode Direction

The ends of selected IPO Curves continue extending in the direction in which they end.

Extend mode Cyclic

The full length of the IPO Curve is repeated cyclically.

Extend mode Cyclic Extrapolation

The full length of the IPO Curve is extrapolated cyclic.

IPO Type

FIXME

Depending on the *active* Object, a **selection** of the following IPO types is available.

Object

Settings, such as the location and rotation, are animated for the *active* Object. All Objects in Blender can have this IPO block. The table below shows the settings that are available for this data block and provides a brief description of each.

IPO Data Elements for Object IPO Data Blocks

IPO Data Item	Description
LocX	Absolute X location
LocY	Absolute Y location
LocZ	Absolute Z location
dLocX	Relative X location
dLocY	Relative Y location
dLocZ	Relative Z location
RotX	Absolute rotation around the X−axis
RotY	Absolute rotation around the Y−axis
RotZ	Absolute rotation around the Z−axis
dRotX	Relative rotation around the X−axis
dRotY	Relative rotation around the Y−axis
dRotZ	Relative rotation around the Z−axis
ScaleX	Absolute scaling along the X−axis
ScaleY	Absolute scaling along the Y−axis
ScaleZ	Absolute scaling along the Z−axis
dScaleX	Relative scaling along the X−axis
dScaleY	Relative scaling along the Y−axis

dScaleZ	Relative scaling along the Z−axis
Layer	Changes in Layer
Time	Changes in time
ColR	Changes to the Red component of the color
ColG	Changes to the Green component of the color
ColB	Changes to the Blue component of the color
FStrength	Strength of Force Fields
FFall	Fall−off of Force Fields
RDamp	Collision Damping
Damping	Random Variation of Collision Damping
Perm	Deflector Permeability − percentage of particles allowed to pass deflector mesh

Material

Settings of the *active* Material are animated for the *active* Object. A numeric field appears immediately to the right of the IPO Type selection field when Material is selected. This field indicates the number of the active Texture *channel*. Eight Textures, each with its own mapping, can be assigned per Material. Thus, per Material−IPO , 8 curves in the row OfsX, OfsY,...Var are available. To change which material you are working on, change the number in the field by left clicking on the field and typing in a new value from 0 to 7.

IPO Data Elements for Material IPO Data Blocks

IPO Data Item	Description
R	Red
G	Green
B	Blue
SpecR	Red component of the specular reflection
SpecG	Green component of the specular reflection
SpecB	Blue component of the specular reflection
MirR	Red component of mirrored light
MirG	Green component of the mirrored light
MirB	Blue component of the mirrored light
Ref	The amount of reflection for diffuse shader
Alpha	Alpha value for the texture
Emit	Sets the amount of light the material emits
Amb	Amount of global ambient color the material receives
Spec	Degree of specularity
Hard	Hardness setting for specular shader

SpTra	SpecTra − makes specular areas opaque on translucent materials
Ior	Index of Refraction
Mode	These are quite a lot of different material settings (I've counted 22 different settings, from shadeless to halo to colorband and so forth).
HaSize	Halo Size
Translu	Translucence − diffuse shading of the back side
RayMir	Amount of mirror reflection for ray trace
FresMir	Power of Fresnel for mirror reflection
FresMirl	*Fac*−Value for Fresnel mirror
FresTra	Fresnel Transparency
FresTral	*Fac*−Value for Fresnel transparency
TraGlow	*Add*−Value for Halo material
OfsX	Fine tunes the X mapping coordinates (Map Input)
OfsY	Fine tunes the Y mapping coordinates (Map Input)
OfsZ	Fine tunes the Z mapping coordinates (Map Input)
SizeX	Sets scaling on the texture's X−axis (Map Input)
SizeY	Sets scaling on the texture's Y−axis (Map Input)
SizeZ	Sets scaling on the texture's Z−axis (Map Input)
texR	Sets Red component of the Map To color
texG	Sets Green component of the Map To color
texB	Sets Blue component of the Map To color
DefVar	Default value the texture uses to mix with other textures
Col	Sets the amount the texture affects color values
Nor	Sets the amount the texture affects the normal
Var	Sets the amount the texture affects other values
Disp	Sets the amount the texture displaces the surface

World

Used to animate a number of settings for the WorldButtons. World too has several texture channels.

IPO Data Elements for World IPO Data Blocks

IPO Data Item	Description
HorR	Red component of the horizon color (world)
HorG	Green component of horizon color (world)
HorB	Blue component of horizon color
ZenR	Red component of the color at the zenith

ZenG	Green component of the color at the zenith
ZenB	Blue component of color at the zenith
Expos	Exponential color correction for the light
Misi	Controls Mist Simulation (mist/stars/physics)
MisDi	Mist Distance (mist/stars/physics)
MisSta	Start of Mist (mist/stars/physics)
MisHi	Height of Mist (mist/stars/physics)
StarR	Amount of red in star color
StarG	Amount of green in star color
StarB	Amount of blue in star color
StarDi	Star distance
StarSi	Star size
OfsX	Offset of texture on x−axis
OfsY	Offset of texture on y−axis
OfsZ	Offset of texture on z−axis
SizeX	Sets scaling on the texture's X−axis (Map Input)
SizeY	Sets scaling on the texture's Y−axis (Map Input)
SizeZ	Sets scaling on the texture's Z−axis (Map Input)
texR	Sets Red component of the Map To color
texG	Sets Green component of the Map To color
texB	Sets Blue component of the Map To color
DefVar	Default value the texture uses to mix with other textures
Col	Sets the amount the texture affects color values
Nor	Sets the amount the texture affects the normal
Var	Sets the amount the texture affects other values

Shape Key

NOTE: Shape keys used to be known as Vertex Keys and Relative Vertex Keys, or RVKs.

Each key applied to the object will be displayed as a field in the right hand portion of the IPO editing window, like the fields for other IPO types. The curves created here are used to control the amount of influence each of the keys has over the shape.

See Absolute Shape Keys and Shape Keys for information on how shape keys are created and how they can be modified using the IPO Data blocks.

Constraint

If the *active* Object has a *constraint* its influence value can be animated via an IPO. Each constraint has its own IPO. Used to display the speed−IPO.

Sequence

The *active* Sequence Effect can have an IPO Curve.

Curve IPO

If the *active* Object is a *path* Curve, this button can be used to display the speed (time) IPO.

Camera IPO

The active camera IPO curves are shown.

Lamp IPO

If the *active* Object is a Lamp, this button can be used to animate light settings, comprehensive of textures.

IPO Menu

FIXME

The DataButtons can be used to control which IPO block is shown and control it.

IPO Window Bar when there is no preexisting IPO data block.

NOTE: When there is no IPO data block associated with an object, the toolbar will look like this. Instead of an IPO data block reference, like you see in the second image, there is only the up/down arrow button, as highlighted in the graphic. To add a data block, just click on the button and select one of the existing data blocks listed, or ADD NEW, if you want to start a new IPO data block.

IPO Window Bar With Existing Data Block

When there is an existing IPO block for an object, the toolbar for the IPO Window will look like the highlighted section in the second graphic.

Pin IPO

The IPO Window shows the current IPO even if the linked Object is deselected.

IPO Menu

Choose another IPO from the list of available IPOs. The option Add New makes a complete copy of the current IPO. This is not visible; only the name in the adjacent button will change. Only IPOs of the same type are displayed in the menu list.

IP :

Give the current IPO a new and unique name. After the new name is entered, it appears in the list, sorted alphabetically.

Users

If this button is displayed, there is more than one user for the IPO block. Use the button to make the IPO "Single User ".

Unlink IPO

The current IPO is unlinked.

Fake User

FIXME

The IPO block is saved even if unused.

Copy to Buffer

All selected IPO Curves are copied to a temporary buffer.

Paste from Buffer

All selected *channels* in the IPO Window are assigned an IPO Curve from the temporary buffer. The rule is: the sequence in which they are copied to the buffer is the sequence in which they are pasted. A check is made to see if the number of IPO Curves is the same.

```
NOTE: If you want to paste IPO data onto an object, the object must
   have an IPO data block to begin with.
```

View Border

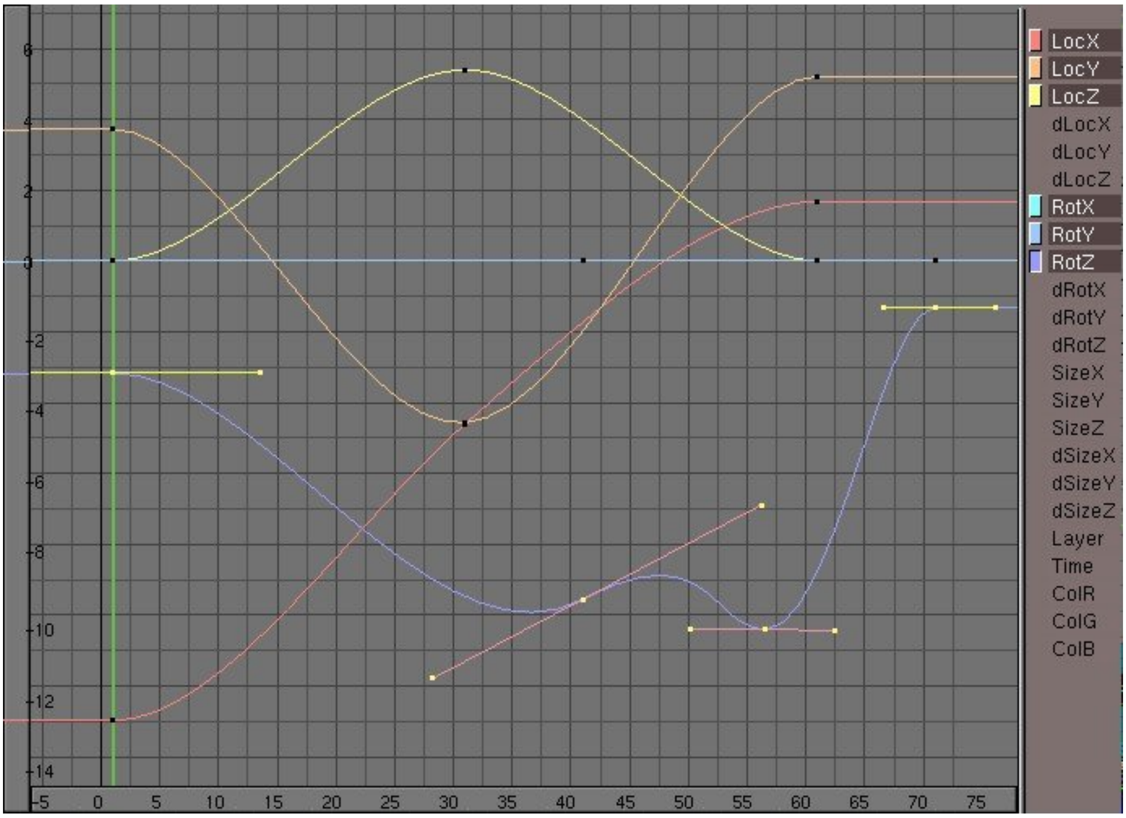
FIXME

Draw a rectangle to indicate what part of the IPO Window should be displayed in the full window.

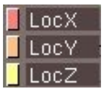
Lock

This button locks the update of the 3DWindow while editing in the IPO Window, so you can see changes made to the IPO in realtime in the 3DWindow. This option works extremely well with relative vertex keys.

IPO Window



FIXME



FIXME

The IPO Window shows the contents of the IPO block. Which one depends on the IPO Type specified in the header. The standard IPO Window has a grid with the time expressed horizontally in frames and vertical values that depend on the *channel*. There are 2 sliders at the edge of the IPO Window. How far the IPO Window is zoomed in can be seen on the sliders, which can also be used to move the *view* .

The right−hand part of the window shows the available *channels* . To make it easier to work with rotation−IPO Curves, they are displayed in degrees (instead of in radiants). The vertical scale relation is: `1.0 'Blender unit' = 10 degrees`. In addition to the IPO Curves, the VertexKeys are also drawn here. These are horizontal blue lines; the yellow line visualises the *reference* Key.

Each *channel* can be operated with two buttons :

IPO Curve Select

This button is only displayed if the *channel* has an IPO Curve. The button is the same colour as the IPO Curve. Use the button to select IPO Curves. Multiple buttons can be (de)selected using SHIFT−LMB .

Channel Select

A *channel* can be selected whether there is an IPO Curve or not. Only IPO Curves of selected channels are drawn. Multiple *channels* can be (de)selected using SHIFT−LMB.

Mouse

CTRL−LMB

Create a new vertex. These are the rules :

- There is no IPO block (in this window) *and* one *channel* is selected: a new IPO Block is created along with the first IPO Curve with one vertex.
- There is already an IPO block, and a *channel* is selected without an IPO Curve: a new IPO Curve with one vertex is added.
- Otherwise a new vertex is simply added to the selected IPO Curve.
- This is *not* possible if multiple IPO Curves are selected or if you are in EditMode.

MMB

Depending on the position within the window :

- On the *channels*; if the window is not tall enough to display them completely, the visible part can be scrolled up and down.
- On the sliders; these can be moved. This only works if you are zoomed in.
- The rest of the window; the *view* is translated.

CTRL−MMB

Zoom in/out on the IPO Window. You can zoom horizonally or vertically using horizontal and vertical mouse movements.

RMB

Selection works the same here as in the 3DWindow: normally one item is elected. Use SHIFT to add/remove from the selection.

- If the IPO Window is in IPO Key mode, the IPO Keys can be selected.
- If at least 1 of the IPO Curves is in EditMode, only its vertices can be selected.
- VertexKeys can be selected if they are drawn (horizontal lines).
- The IPO Curves can be selected.

RMB and drag

Select and start *translation* mode, i.e. the Grabber. The selection can be made using any of the four selection methods discussed above.

SHIFT−RMB

Adds/removes from the selection.

HotKeys

NUM− NUM+

Zoom in, zoom out.

PAGEUP

Select the next IPO Key. If more than one IPO Key is selected, the selection is shifted cyclically.

SHIFT−PAGEUP

Add the next IPO Key to the selection.

PAGEDOWN

Select the previous IPO Key. If more than one Object Key is selected, the selection is shifted cyclically.

SHIFT−PAGEDOWN

Add the previous IPO Key to the selection.

HOME

All visible curves are displayed completely, centered in the window.

TAB

All selected IPO Curves go into or out of EditMode. This mode allows you to transform individual vertices.

A

Select All / deselect All. If any item is selected, first everything is deselected. Placing the mouse cursor above the *channels*, (de)selects all channels where there is a curve.

B

Border select. Draw a rectangle with the LMB; all items that fall within this rectangle are *selected*. Draw a rectangle with the RMB to *deselect* .

SHIFT−C

Centers the view on the current *frame*, without zooming.

C

If one vertex or one IPO Key is selected, the current *frame* number is set to this position.

SHIFT−D

Duplicate IPO. All selected vertices or IPO Keys are copied. Then *translation* mode is started automatically.

G

Translation mode (the Grabber). This works on selected curves, keys or vertices. Alternatives for starting this mode :

- RMB and drag.

The following options are available in *translation* mode:

- Limiters:
- CTRL increments of 1 frame or vertical unit.
- SHIFT−CTRL increments of 0.1 frame or vertical unit.
- MMB restricts the current translation to the X or Y axis. Blender calculates which axis to use, based on the already initiated mouse movement. Click MMB again to restore unlimited translation.
- ARROWS: With these keys the mouse cursor can be moved exactly 1 pixel.
- X Limits translation to the X axis.
- Y Limits translation to the Y axis.

Grabber can be terminated with:

- LMB, SPACE or ENTER: Move to the new position.
- RMB or ESC: Everything returns to the old position.

H

Toggle Handle *align* / *free* .

SHIFT−H

Set Handle *auto*. The selected Bezier *handles* are converted to *auto* type.

I

Insert Key. Vertices can be added to the visible curves in the IPO Window. A PopupMenu asks you to make a choice :

- Current Frame: All visible curves get a vertex on the current frame.
- Selected Keys: (only in IPO Key mode) all selected IPO Keys get vertices for each visible curve, including IPO Curves that are not part of the IPO Key.

J

Join vertices. Selected vertices or IPO Keys can be joined. A PopupMenu asks you to make a choice :

- All Selected: all selected vertices are replaced by a new one.
- Selected doubles: all selected vertices that are closer to each other than 0.9 *frame* are joined.

K

IPO Key mode ON/OFF. If the IPO block is Object IPO type, the Objects are redrawn with the option DrawKey ON (see the explanation under IPO Header).

R

Recording mode. The X and Y movements of the mouse are linked to the height of the IPO Curve. Thus, this works with a maximum of two selected *channels* or IPO Curves. The old curve is completely deleted; the new one becomes a 'linear' type. You cannot change parts of curves with *recording*. The scale at which this happens is determined by the *view* of the IPO Window. A PopupMenu asks you to make a choice :

- Still: The current frame is used as the starting point.
- Play anim: The animation starts, allowing you to see the correlation with other animation systems.

During *recording* mode, the CTRL must be held down to actually start recording. Press SPACE or ENTER or LMB to stop *recording*.

Use ESC to undo changes.

S

Scaling mode − This works on selected IPO Curves and vertices. The degree of *scaling* is *precisely* linked to the mouse movement. Try to move from the (rotation) midpoint with the mouse.

In IPO Key mode, you can only *scale* horizontally.

Limiters :

- CTRL: in increments of 0.1.
- SHIFT−CTRL: in increments of 0.01.
- MMB limits *scaling* to the X or Y axis. Blender calculates which axis to use based on the already initiated mouse movement. Click MMB again to return to free *scaling*.
- ARROWS: These keys allow you to move the mouse cursor exactly 1 pixel.
- X Limits *scaling* to the X axis.
- Y Limits *scaling* to the Y axis.

Terminate size mode with:

- LMB SPACE or ENTER: To finalize scaling.
- RMB or ESC: Everything returns to its previous state.

SHIFT−S

Snap Menu.

- Horizontal: The selected Bezier handles are set to horizontal.
- To next: The selected handle or vertex is set to the same (Y) value as the next one.
- To frame: The selected handles or vertices are set to the exact frame values.
- To current frame: The selected handle or vertex is moved to the current frame.

T

If an IPO Curve is selected: "IPO Type". The type of selected IPO Curves can be changed. A PopupMenu asks you to make a choice :

- Constant: After each vertex of the curve, this value remains constant, and is not interpolated.
- Linear: Linear interpolation occurs between the vertices.
- Bezier: The vertices get a *handle* (i.e. two extra vertices) with which you can indicate the curvature of the interpolation curve.

If a Key is selected: "Key Type". The type of selected Keys can be changed.

- Linear: Linear interpolation occurs between the Keys. The Key line is displayed as a broken line.
- Cardinal: Fluent interpolation occurs between the Keys; this is the default.
- BSpline: Extra fluent interpolation occurs between the Keys, four Keys are now involved in the interpolation calculation. Now the positions *themselves* cannot be displayed precisely, however. The Key line is shown as a broken line.

V

Vector Handle. The selected Bezier *handles* are converted to *vector* type.

X

Erase selected. The selected vertices, IPO Keys or IPO Curves are deleted. If there are selected VertexKeys, they are also deleted.

See also

- PART VII − ANIMATION BASICS

Previous: Reference/Windows/3D	Contents	*Next:* Reference/Windows/Sequence

 Introduction To Blender Constraints

ight:100%;">

[[{{{2}}}|]]>

ight:100%;">

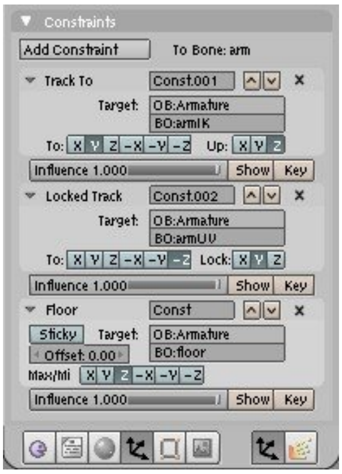
[[{{{3}}}|]]>

Blender Summer of Documentation: Contents | Manual | Blender Version **2.42**

Introduction To Blender Constraints

Constraints are object features that define spatial relationships between objects, and are the standard method for controlling characters among all 3D animation packages that still implement a more traditional approach to digital character animation. In Blender, constraints can be associated to any type of object or bone object, but not all constraints work with bone objects, and not all constraints work with normal world objects.

General Constraint Properties



Constraints are accessed via the object buttons (F7) in the Constraints panel. After you press the Add Constraint button and select the desired constraint type from the menu, a constraint UI is added to the panel. The constraint will be linked to the active object or the active selected bone, indicated by the label To Object: or To Bone: on the right of the Add Constraint menu.

Constraints are evaluated in a specific order: from top to bottom of the constraint stack. This order can be viewed and changed inside the Constraints panel. Each constraint has a pair of arrow buttons on the right of the constraint name in its top right corner, which can be used to move the constraint up or down the constraint stack. The little cross on the right of these buttons can be used to delete the constraint from the stack.

NOTE: The name field of a newly added constraint will appear red, which indicates that the constraint is not functional. In the case of a newly added constraint this is because the constraint does not yet have a target specified. All constraints require a target object (a normal world object or an armature bone). You should enter the name of the desired target object in the Target: OB field. When you want an armature bone as target, enter the name of the armature object as target. A new field with BO: will appear, where you can place the name of the target bone.

The name field of a constraint will also turn red if the settings for the constraint are invalid or if the constraint conflicts with another constraint. For example: A Track To constraint of which the To and Up vector are set to Z.

Influence

The **influence** of a constraint on the actual location/rotation/size of the object can be defined with the Influence value slider. This can be linked an Ipo Curve, which can be evoked in the Ipo Curve Editor with the button Show on the right of the Influence value slider. The Key button beside it can be used to add a key to the Ipo Curve.

You can key any constraint, and an object can have multiple constraints keyed to exert a different influence at different times.

Multiple Influence Example

Suppose that you want a camera to follow one path for awhile and then drift off and follow another path. You apply a *follow path* constraint to a camera for one circle/curve/path (make sure you have curve path enabled and enter its name), then open up an IPO curve window. In the buttons window (constraint panel) you can either hit I and insert an Influence value by sliding the slider (to vary the force strength or force fall off) or LMB click the "key" button at the bottom of the constraint panel. Once you have inserted the initial key, up arrow or change to the frame where you want the first influence to start falling off. Go to the IPO window and and Ctrl LMB to insert a key. Go forward a few frames, where you want its influence to be zero, and insert another key, and edit it so that its Y value is zero. At that frame, the constraint in the stack will not influence the camera motion at all. Now insert a *second* Follow Path constraint, and enter the name of the *second* path. Now position back a few frames and insert a key to where you want its influence to start (probably where the first constraint starts dropping off). Edit this second influence curve to come up from zero to one over time. You can imagine that the IPO curves for each circle should cross over each other so that when one is at full influence (max 1.0) the other is at zero etc. You might have to play with the curves a bit to get the transition smooth.

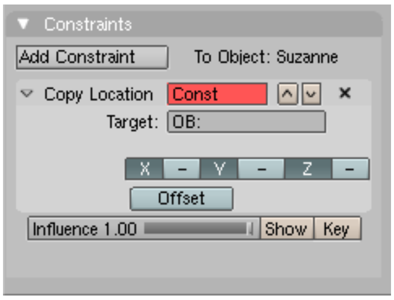
Constraints on Bones

Concerning bones with constraints: the color of the bones in the 3D Window indicate what type of constraint is used:

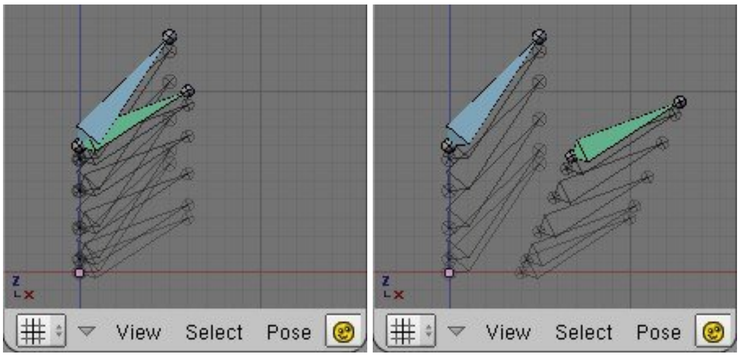
- **Grey**: No constraint.
- **Yellow**: A bone with an IK constraint.
- **Orange**: A bone with an IK constraint but no target.
- **Green**: A bone with any other kind of constraint.
- **Blue**: A bone that is animated with keyframes.
- **Purple**: The Stride Root.

Constraint Types

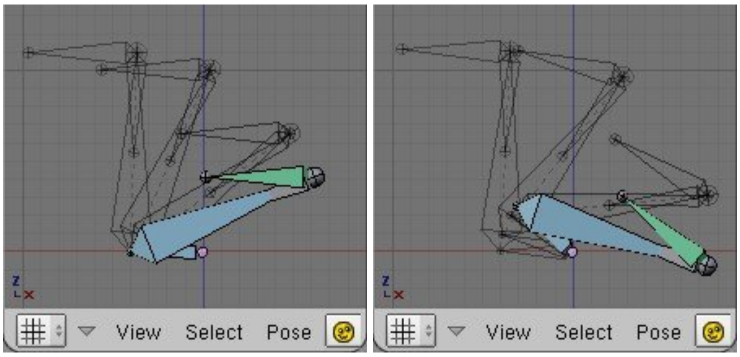
Copy Location



Copy Location forces the object to have the same location as its target. When this constraint is used on a bone and another bone is the target, a new button−−labeled **Local**−−appears next to the **X Y Z** buttons. Using the local button causes the local translation values of the target to be given to the constrained object. In rest position, all bones are at the local location of (0,0,0).

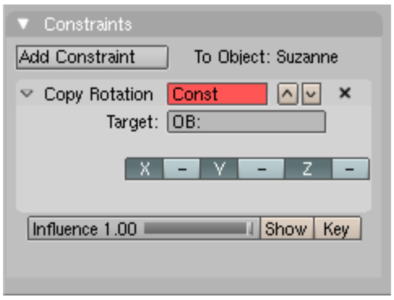


Left: Using global space. This is the default behavior; it's what happens when the local button is not activated. Right: Using local space, with local button activated.

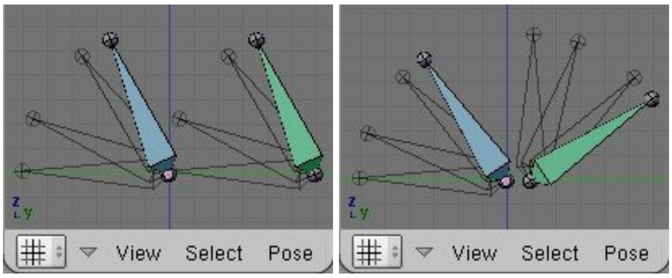


In these last two images, the constrained bone (shown in green) is actually a child of the root bone (the bone at the beginning of the chain). This demo shows possible uses for the location constraint in a rig. Note that the green bone still inherits rotation from the root because the root is its parent. This is by design though; the green bone could be the child of any of these bones, or none of them.

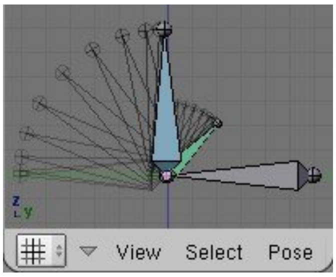
Copy Rotation



Copy rotation causes one object to match the rotation of a target object. For bones, a local option will appear, allowing you to use local space. Imagine you have a bone pointing up and a bone pointing down. If you use local space, each can point different directions, but when the target bone moves to its left, the affected bone will move it to *its* left.

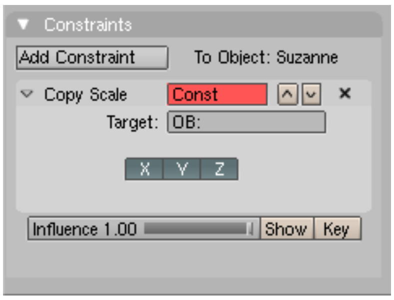


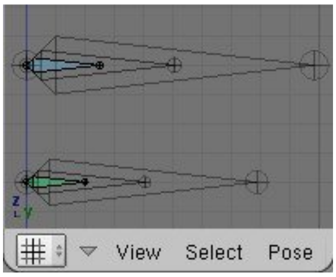
Left: Using global space. Right: Using local space.



Here is one good use of the rotation constraint and local space. By setting the influence value to 0.5, the small bone will rotate half as far as the target. This is useful for character joints.

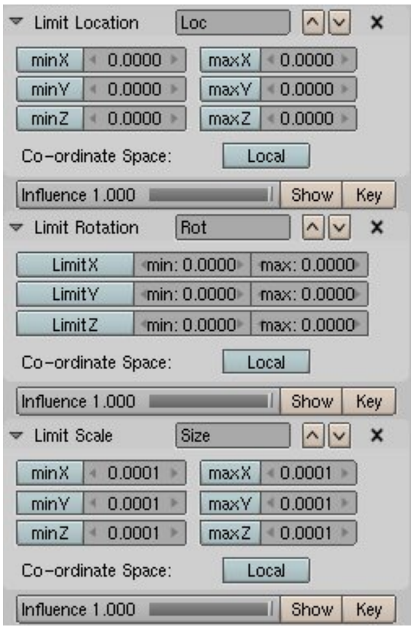
Copy Scale





Copy Scale forces the affected object to have the same size as the target. All bones have a (1,1,1) size in rest position. We can draw a bone that is 10 million times longer than the bone right next to it, but in pose mode, they both have a starting size of (1,1,1). You should keep this in mind if you're going to be using the copy scale constraint.

Limit Constraints



Limit Loc/Rot/Size constraints

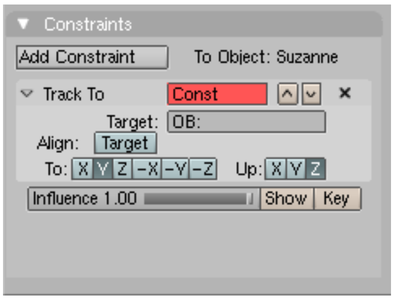
These constraints limit the scale, rotation, or location of the active object. The limit can be named, and is always specified in XYZ terms. Each limit min or max must be enabled by LMB, turning it dark green.

Limit Location
: The object cannot be located or positioned below the minimum, or above the maximum amount. Use this to constrain an object to only move within a confined space.

Limit Rotation
: The object cannot be rotated below the minimum, or above the maximum amount. Use this to constrain a bone, for example, not to hyperextend.

Limit Scale
: The object cannot be scaled below the minimum, or above the maximum amount.

Track To



Track To rotates an object to point at a target object. This constraint also forces the object to be "right−side" up. You can choose the positive direction of any of the three axes to be the up−side. This constraint shares a close relationship to the IK constraint in some ways. This constraint is very important in rig design, and you should be sure to read and understand the page on tracking, as it centers around the use of this, and the IK, constraints.

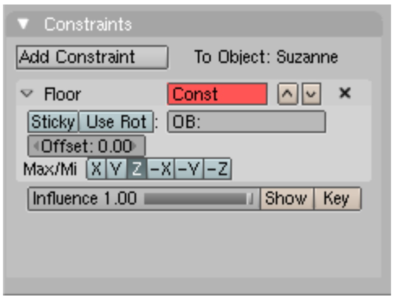
Settings

To
: The axis of the object that has to point to the target.

Up
: The axis of the object that has to be aligned (as much as possible) with the world Z axis. An Align: Target button, when enabled, uses the coordinates of the target object's Z axis, thus tilting or rocking the object as it tracks the target.

Head/Tail − With Bone targets only
: A number from 0.0 to 1.0 that represents the place on the target bone to Track to (0.0 = the bone's root; 1.0 = the bone's head)

Floor



Animation Tip:

When you animate foot placement on the floor plane, ***always*** be sure to use the option **VisualLoc** from the **Insert Key** menu, or enable **Use Visual Keying** from the **Auto Keyframing** menu in the User Preferences.

The Floor Constraint allows you to use a target object to specify the location of a plane which the affected object cannot pass through. In other words, it creates a floor! (or a ceiling, or a wall). This only works by default with planes of the global coordinate system.

Settings

Sticky
: Makes the affected object immovable when touching the plane (cannot slide around on the surface of the plane), which is fantastic for making walk and run animations.

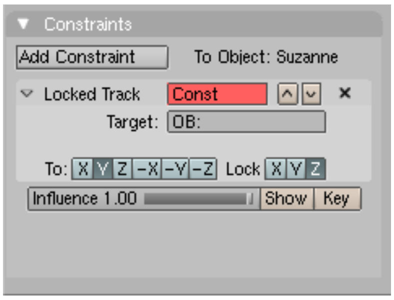
UseRot
: Takes the target object's rotation into account.

Offset
: A number of BU's from the object's center. Use this to account for the distance from a foot bone to the surface of the foot's mesh.

Max/Min
: Which axis is the floor? Things normally stick "down" Z, but can walk on walls as well.

Head/Tail − With Bone targets only
: A number from 0.0 to 1.0 that represents the place on the target bone to use for the Floor (0.0 = the bone's root; 1.0 = the bone's head)

Locked Track



Locked Track is a difficult constraint to explain, both graphically and verbally. The best real−world example would have to be a compass. A compass can rotate to point in the general direction of its target, but it can't point *directly at* the target, because it spins like a wheel on an axle. If a compass is sitting on a table and there is a magnet directly above it, the compass can't point to it. If we move the magnet more to one side of the compass, it still can't point *at* the target, but it can point in the general direction of the target, and still obey its restrictions of the axle.

When using a Locked Track constraint, you can think of the target object as a magnet, and the affected object as a compass. The "Lock" axis will function as the axle about which the object spins, and the "To" axis will function as the compass needle. Which axis does what is up to you! If you have trouble understanding the buttons of this constraint, read the tool−tips; they're pretty good. If you don't know where your object's axes are, turn on the **Axis** button in the **Draw** panel, object buttons, F7. Or, if you're working with bones, turn on the **Draw Axes** button, **Armature** panel, editing buttons, F9.

This constraint was designed to work cooperatively with the Track To constraint. If you set the axes buttons right for these two constraints, Track To can be used to point the axle at a target object, and Locked Track can spin the object around that axle to a secondary target.

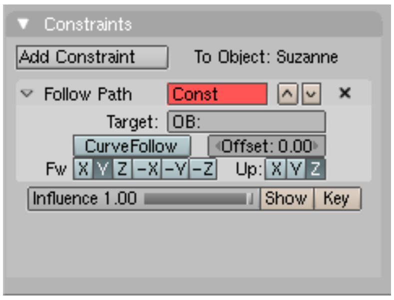
This is all related to the topic discussed at length in the Tracking section.

Settings

To
: The axis of the object that has to point to the target.

Lock
: The axis of the object that has to be locked

Follow Path



Follow Path places the affected object onto a curve object. Curves have an animated property that causes objects along the path to move. In order for this to work, you have to have the CurvePath option activated (it's a property of the curve object). You can find it in the Curve and Surface panel in the Editing buttons, F9.

The movement along the path might be controled by two different ways: the most simple, in this same Curve and Surface panel, is to define the number of frames of the movement *via* the num button Path Len:, and its start frame *via* the constraint's Offset: option (by default: start frame **1** (= offset of **0**), duration **100**).

The second way – much more precise and powerful – is to define a Speed IPO curve for the path (Path section of the IPO Curve window). The start position along the path will correspond to an IPO value of **0.0**, and the end position, to an IPO value of **1.0**. You can therefore control the start frame, the speed of the movement, and the end frame, and even force your object to go forth and back along the path!

If you don't want objects on the path to move, you can give the path a flat speed IPO curve (its value will control the position of the object along the path).

Follow Path is another constraint that works well with Locked Track. One example is a flying camera on a path. To control the camera's roll angle, you can use a Locked Track and a target object to specify the up direction, as the camera flies along the path.

This constraint does not work well with bones.

Settings

Offset
: The number of frames to offset from the "animation" defined by the path (by default: from the frame **1**).

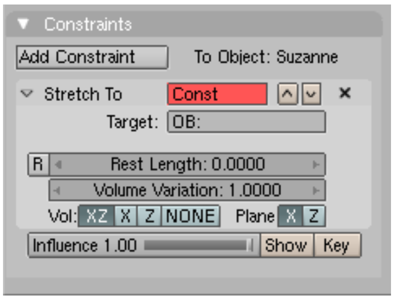
CurveFollow
: If this option isn't activated, the affected object's rotation isn't modified by the curve; otherwise, it's affected depending on the following options:

Fw
: *The axis of the object that has to be aligned with the forward direction of the path.*

Up

The axis of the object that has to be aligned (as much as possible) with the world Z axis.
In fact, with this option activated, the behaviour of the affected object shares some properties with the one caused by a Locked Track constraint, with the path as "axle", and the world Z axis as "magnet"…

Stretch To



Stretch To causes the affected object to scale the Y axis towards a target object. It also has volumetric features, so the affected object can squash down as the target moves closer, or thin out as the target moves farther away. Or you can choose not to make use of this volumetric squash−'n'−stretch feature, by pressing the **NONE** button. This constraint assumes that the Y axis will be the axis that does the stretching, and doesn't give you the option of using a different one because it would require too many buttons to do so.

This constraint affects object orientation in the same way that Track To does, except this constraint bases the orientation of its poles on the original orientation of the bone! See the page on Tracking for more info. Locked Track also works with this constraint.

Settings

R

Pressing the R button calculates the rest length as the distance from the centers of the constrained object and its target

Rest Length

Rest Length determines the size of the object in the rest position

Volume Variation

Volume Variation controls the magnitude of the effect

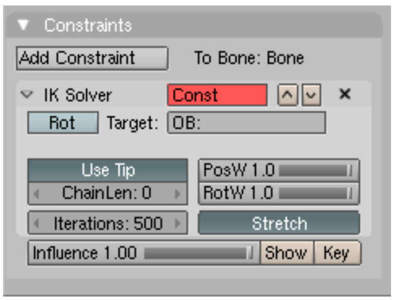
Vol

The Vol: buttons control along what axes the volume is preserved (if at all)

Plane

The Plane buttons define which local orientation should be maintained while tracking the target

IK Solver

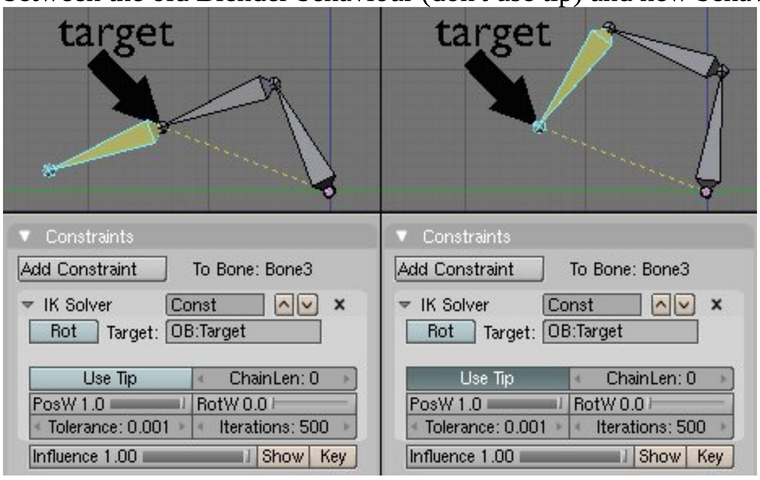


The IK Solver constraint is Blender's implementation of inverse kinematics. You add this constraint to a bone and then it, and the bones above it, become part of the inverse kinematic solving algorithm.

Settings

Rot
: Toggle option to make the IK chain follow the rotation of the target object.

Use Tip
: Toggle option to use the tip of the bone instead of the base to solve the IK to. This option toggles between the old Blender behaviour (don't use tip) and new behavior (use tip).



An IK constraint on the yellow bone with indicated target. Left: without Use Tip. Right: with Use Tip.

ChainLen
: The number of bones above this bone that you want to be affected for IK. The default is 0, which means *all* bones above this bone will be used for IK.

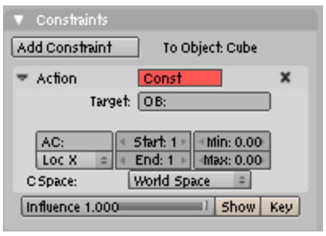
PosW
: foo

RotW
: foo

Tolerance
: foo

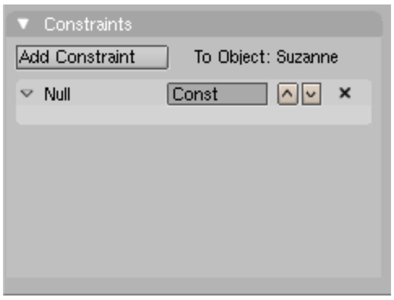
Iterations
: foo

Action



The Action constraint allows you to map any action to one of the rotation axes of a bone.

Null



The null constraint doesn't do anything. It is an antiquated feature.

Previous: BSoD/Introduction_to_Rigging/The_Armature_Object Contents BSoD/Introduction_to_Rigging/Preface:_Rigging_i

User Manual: Contents | Guidelines | Blender Version **2.4x**

Hooks are similar to Shape Keys in that they deform a mesh over time (frames). The difference is that hooks make it look like the mesh is snagged with a fish hook. Moving the hook moves selected vertices under the influence of the hook (which is really just an Empty), and you make the hook move by animating the motion of the empty through Ipo keys. As the hook moves, it pulls weighted vertices from the mesh with it. If you have used Proportional Editing, you can think of it as animated proportional editing. While hooks do not give you the fine control over vertice movement that Shape Keys do, they are much simpler to use.

Object Hooks

Mode: Object Mode / Edit Mode

Panel: Editing Context Modifiers

Hotkey: Ctrl H

Menu: Mesh Vertices Add Hook

Description

Hooks give access at object level to the underlying geometry of meshes, curves, surfaces or lattices. A hook is an object feature and it is like a parent to an object, but for vertices. You can create as many hooks to an object as you like, and assign for each hook vertices that will be affected. Overlapping hooks is also possible, here a weighting factor per hook is provided that determines the amount each hook will affect the overlapping vertices.

All object level options and transformations are possible on the hook object, including using hierarchies, constraints, ipo and path animations. You can also make the hook−parent a child of the original object if you don't want object transformations to deform the hooks.

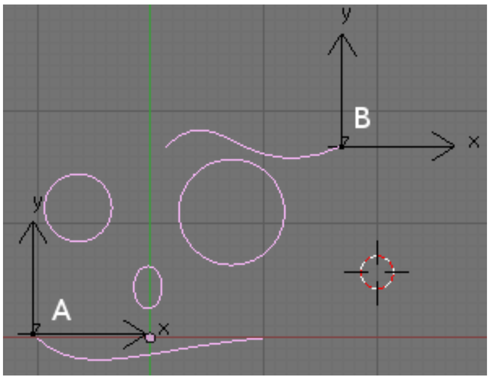
Note

When you change topology (i.e. more destructive editing than just manipulating existing points), you most likely have to reassign existing hooks as well.

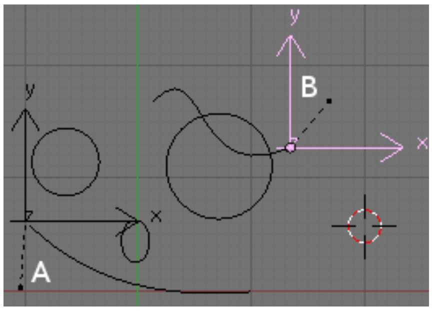
Examples

A typical example of using Hooks is to animate vertices or groups of vertices. For example, you may want to animate the vertices associated with a "Mouth" on a character's face.

In (*Animated face frame 1*) and (*Animated face frame 10*) a face made from Bezier curves has two Hooks applied. One applied to a control−point on the mouth labeled "**A**" and one applied to the eyebrow labeled "**B**". The animation has 10 frames over which Hook **A** moves up and Hook **B** moves down.



Animated face frame 1.



Animated face frame 10.

Adding Hooks

Mode: Edit Mode

Panel: Editing Context Modifiers

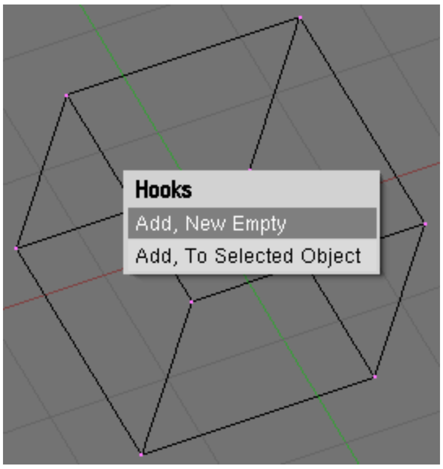
Hotkey: Ctrl H

Menu: Mesh Vertices Add Hook

Description

Since hooks relate to vertices or control points, most editing options are available in edit mode for meshes, curves, surfaces and lattices.

Options



Hooks menu

Add, New Empty
: Adds a new hook and create a new empty object, that will be a parent to the selection, at the center of the selection

Add, To Selected Object

When another object is selected (you can do that in edit mode with Ctrl RMB) the new hook is created and parented to that object

Editing Hooks

Mode: Edit Mode

Panel: Editing Context Modifiers

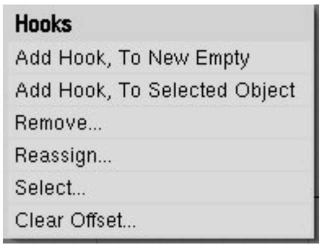
Hotkey: Ctrl H

Menu: Mesh Vertices Add Hook

Description

Once hooks are available in an object, the hook menu will give additional options:

Options



Hooks extended menu

Remove
: This will give a new menu with a list of hooks to remove

Reassign
: Use this if you want to assign new vertices to a hook

Select...
: Select the vertices attached to a specific hook

Clear Offset...
: Neutralize the current transformation of a hook parent

Hook Modifier

Mode: Object Mode / Edit Mode

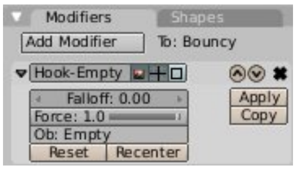
Panel: Editing Context Modifiers

Hotkey: Ctrl H

Description

Hooks are modifiers, that are added to the modifier stack. For each hook modifier, you can give a hook a new name, the default name is the parent name, give it a new parent by typing the new parents name or assign it a Force weighting factor.

Options



In the editing buttons, modifier panel, when a hook is created, you can control it via the panel.

Ob
: The parent object name for the Hook. Changing this name also recalculates and clears offset

Reset
: Recalculate and clear the offset transform of Hook

Recenter
: Set Hook center to cursor position

Select
: Select affected vertices on mesh

Reassign
: Reassigns selected vertices to this hook

Force
: Since multiple hooks can work on the same vertices, you can weight the influence of a hook this way. Weighting rules are:
    - ◊ If the total of all forces is smaller than 1.0, the remainder, 1.0−forces, will be the factor the original position have as force.
    - ◊ If the total of all 'forces' is larger than 1.0, it only uses the hook transformations, averaged by their weights.

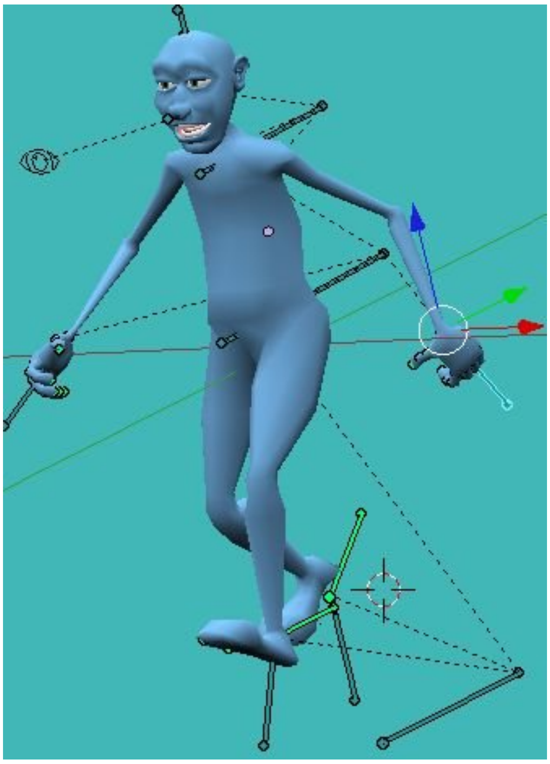
Falloff
: If not zero, the falloff is the distance where the influence of a hook goes to zero. It currently uses a smooth interpolation, comparable to the Proportional Editing Tools. (See *mesh_modeling_PET*)

Previous: Manual/Explode Modifier	Contents	*Next:* Manual/Lattice Modifier

User Manual: Contents | Guidelines | Blender Version **2.45**

Character Animation using the Armature System

As we have seen in Your First Animation in 30 + 30 Minutes Part II Blender uses *Armatures* for character animation. An armature, once parented to our character mesh, is just like a skeleton which will let us define a number of *poses* for our character along the timeline of our animation. An armature is made up of an arbitrary number of *bones*. The size, position and orientation of every bone in your armature is up to you, and you will find through this chapter that different situations will require a particular arrangement of bones for your character to work properly.



As you animate your armature you will find that it is better to organize several related poses in something called an *action*, which is more or less the same as in the real world. When we walk, we can imagine ourselves passing through several instantaneous poses as if we were in the frames of a moving picture, the whole process of the walk is an action in the end.

But there are actions and actions. As an animator you will need to acquire the capability of knowing how to split any natural movement or action into several simpler actions that will be easier to deal with. Working with simpler actions commonly saves time and work (and why not: money!) since these actions are usually reusable.

Once you have set−up your first actions you will be able to combine them using Blender's powerful *Non Linear Animation* (or *NLA*) editor, giving your character a living mood and natural manners.

In this chapter we will cover every single detail of Blender's functionalities related to Armatures, Actions and the NLA Editor. Furthermore we will see several armature set−ups that will give you a starting point for your own creations and ideas. Relax and enjoy.

Chapters

Previous: Manual/Lattice Animation	Contents	*Next:* Manual/Armature Objects

User Manual: Contents | Guidelines | Blender Version **2.45**

Adding an Armature

Mode: Object Mode / Edit Mode (Armature)

Hotkey: Shift A

Menu: Add Armature

Description

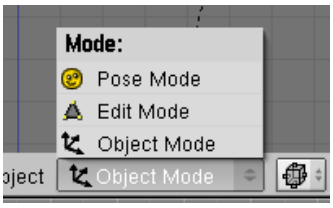
Armatures are like articulated skeletons, that allow you to pose and deform the geometry that surrounds it. An armature is made of a series of bones connected to each other via parenting or constraints. Armatures are most often used in character animation, as a manipulatable skeleton to pose a character, but they can also be useful in other situations too.

An armature is like any other object type:

- It has a center, a position, a rotation and a scale factor.
- It has ObData that can be edited
- It can be linked to other scenes, and the same armature data can be reused on multiple objects.
- All animation you do in object mode is only working on the object, not the armature's contents like bones.

An armature has 3 modes. You can switch using the dropdown menu in the header of the 3Dview or use Tab to switch between Armature Edit Mode and Object Mode. When in ObjectMode, you can switch the Pose Mode on and off with Ctrl Tab. Rather than being an explicit mode that excludes all others, Pose Mode is a state of the armature you can switch on or off. So when in PoseMode, you are still in ObjectMode (you can select another object, contrary to the EditMode).

Options



Mode Pulldown

Object Mode
: Your armature is like any other Object, you can move it around the scene, scale it, rotate it and edit options in the Object button window (F7).

Edit Mode
: Your armature is in what we call rest position, you can move, add and delete the bones it contains.

Pose Mode
: Your armature is ready to be animated, each bone can be moved, scaled or rotated relative to the rest position defined in Edit Mode. Constraints can be applied, you can pose your character, add keys and animate the bone's behavior over time.

Armature Datablock

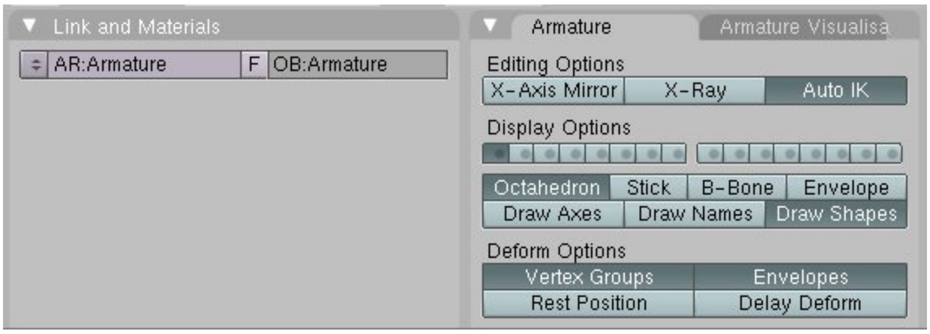
Mode: Object Mode / Edit Mode (Armature)

Panel: Editing Context Link and Materials

Description

Armatures consist of an Object and an Armature data block. These can be manipulated in the same way as other objects

Options



Armature Panel

AR
: Rename the armature Datablock. The dropdown is a quick way to select which Armature datablock you want to connect to this armature object. You can keep more than one version for the same character. Useful when you have a special move to achieve in a shot, you can turn on an armature for special purposes.

F
: Assign a fake user to the Armature. Again, if you have more than one armature for your character, it's a good idea to turn the Fake on, as if your armature datablock is not used (linked) it's not going to be saved in your `.blend` files. You can always do batch fake−assignement of armatures by opening the Datablock browser (Shift F4), go back one level (..) to see the root of your project, then go in Armature datablock, select all the armatures you want to keep and press the F.

OB
: Rename your armature object to something more cool and useful than `Armature`, `Armature.001`, etc...

Armature Display Options

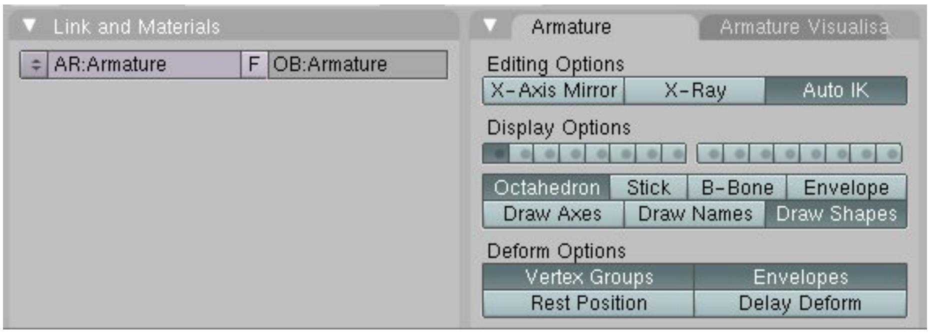
Mode: Object Mode / Edit Mode (Armature)

Panel: Editing Context Armature

Description

Various methods to control how Armatures are displayed in the 3D View are available. Also note there are some specific options and features that relate to the display mode you're in.

Options



Armature Panel

X−Ray
: The same as in the Object Context, this lets you see the armature through anything in the scene, solid or not. It's useful to see where your bones are in your character so you can select them.

Display Options

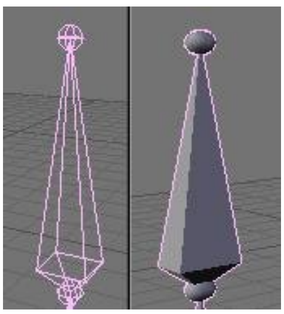
Much like Object Layers, an Armature element (bone, for example) can exist on one or more layers. To work on only those bones of interest to you, select which layers to display and unclutter your view.

Octahedron
: This is the default view. Nothing exciting except you have a good idea of the rolling of the bones.

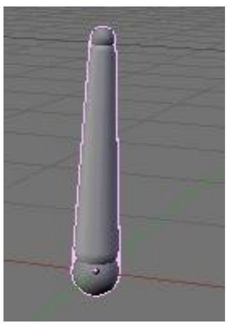
Stick
: This display mode is really useful when you have a lot of bones in your view. It lets you "unclutter" the screen a bit. It draws the bones as tiny sticks.

B−Bones
: It's more a feature than a display mode. This is useful to visualise the effect you get when you activate the B−bones (Bezier−Bones). Each join between 2 bones acts like a curve handle and lets you get extremely curvy poses. This is explained further in the Posing Armatures section.

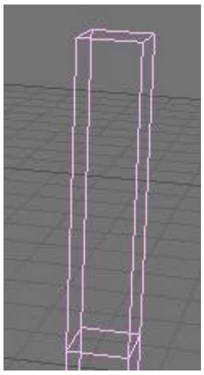
Envelope
: Again it's more a feature than a display mode, using envelopes lets you deform all vertices within a bone's proximity, rather than having to explicitly define and weight paint vertex groups. In this case the visualization will be useful to tweak your rig, showing you which part of the mesh that a bone will move. It's possible interactively change the zone of influence in this display mode. The zone is only visible in EditMode or PoseMode though.



Stick

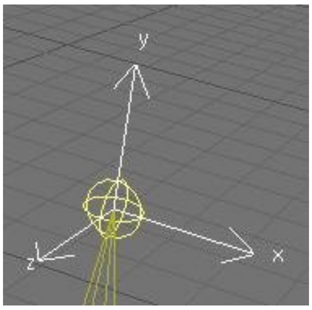


Octahedron



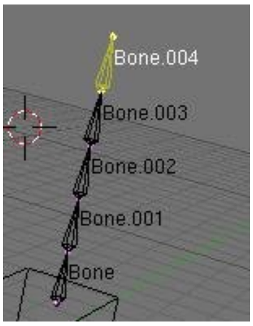
Envelope

B−Bones



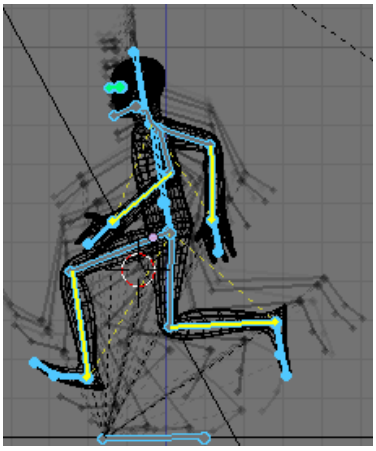
Draw Axes

Draw Axes
: To draw the axes on each bone of the armature when you are in Editmode of PoseMode. Handy when you want to know where you are, and which axis to use in a constraint for example. Mental note: Y is up, Z is depth and X is side, contrary to object for which Z is up, Y is depth and X is side. (*Draw Axes*)



Draw names

Draw names
: This lets you see names of bones whatever the mode you are in. It's useful again to edit your armature, create parent dependencies or add constraints. (*Draw names*)



Ghost

Ghost
: Shows a transparent 'ghost' of the armature N frames behind and over the current time. This only works when you have an action linked to the armature, as we will see in the Posing Armatures section.(*Ghost*)

Step
: The frame interval between ghost instances.

Armature Deformation

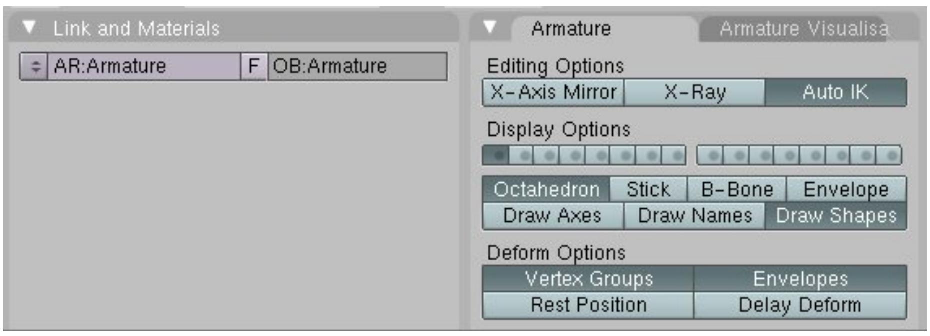
Mode: Object Mode / Edit Mode (Armature)

Panel: Editing Context Armature

Description

Armatures can deform meshes either through a parent relationship or through modifiers. There are various options to control this.

Options



Armature Panel

Vertex Groups & Envelopes
: Those two toggles let you choose if you want the armature to deform your character using the Vertex Groups and/or the Envelopes. These are used when the Armature is deforming via a parent

relationship. If you are using an Armature modifier, these controls are available per modifier in the modifiers panel.

Rest position
: This will show the character as factory default (as defined in EditMode), no actions will be applied to the armature so you can easily edit it in the middle of an animation.

Delay Deform
: This was useful before when the old system was *very* slow. What it does is when you do a manipulation to the rig, it waits until you finish to update the view.

Previous: Manual/Character Animation	Contents	*Next:* Manual/Editing Armatures

User Manual: Contents | Guidelines | Blender Version **2.45**

Editing Armatures

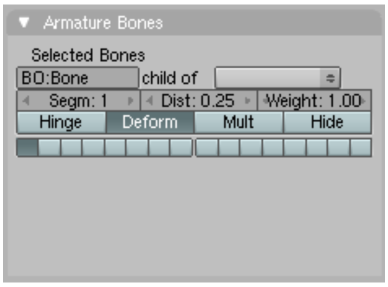
Mode: Edit Mode (Armature)

Panel: Editing Context Armature

Description

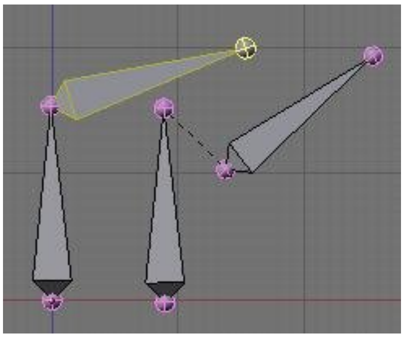
Armatures are comprised of Bones. Editing an Armature in Edit Mode allows you to manipulate the bones in their default rest position.

Options



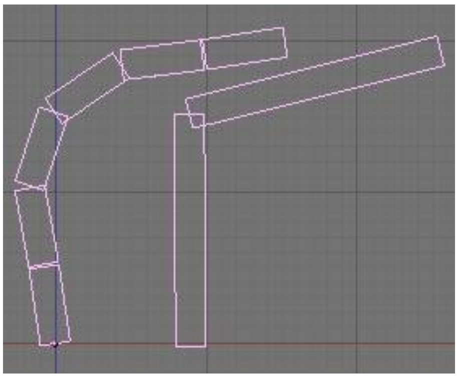
Edit panel: Bones.

BO:
: Rename your bone.



Child of

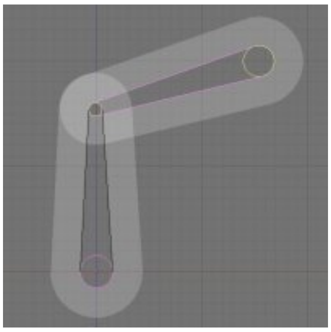
Child of
: Dropdown: lets you choose which bone will be the parent of this bone. If so there will be a small button "co" meaning connected. This will just set the relationship between your bones. If you just parent it to another bone, it will do everything the parent does, rotate, move and scale. A dotted line between the parent and child will appear. If you select Connected, the Root of the Children will go stick to the tip of the parent, giving you a chain of bones like the 2 bones in your arm. (*Child of*)



Segmented Bone

Segm

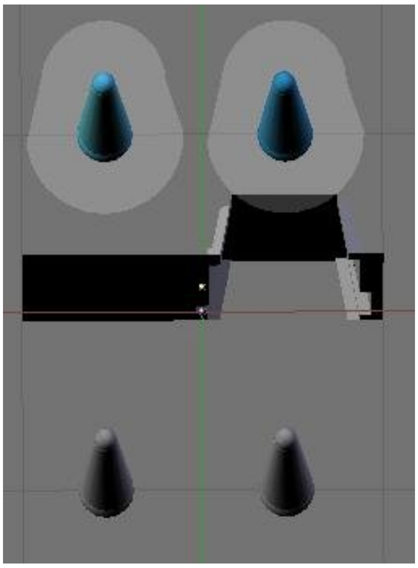
This is the option to use B−Bones. Set to something > 1, it will cut your bone in many little segments and will deform them on a bezier curve. It really show off in a chain of bones though. See the example between 1 segment (right) and 3 segment each (left). (I returned in Object mode to see the effect). (*Segmented Bone*)



Envelope

Dist

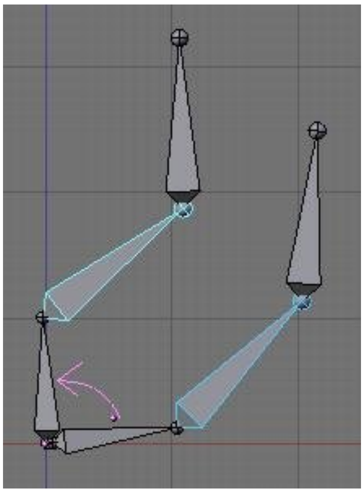
This is the area of influence of the bone. it can be visualized using the Envelope display mode. We generally don't touch this field as there is a more easy and fast way to change this option. Turn Envelope drawmode on and select a bone. Then using Alt S, you can scale the zone of influence. Better: you can do it on multiple bones and you can do it in EditMode and PoseMode. (*Envelope*)



Weight

Weight

This gives you the liberty to tell if this bone will work more or less on the geometry. For example if you have 2 bones working with envelope and crossing each other, you can tell one of them to get more power by lowering the weight of the bone you don't want to use much. If both bones have the same weight (like 1:1) they will influence the geometry equally. But if you set one to 0.5. The other one at 1 will influence more. For example in this image, 2 bones using envelope influence try to move the same geometry. The 2 on the left have the same weight, you can see the geometry didn't move. On the right one of the bones has 0.5 so the bone with 1 of weight is winning the pulling contest!. (*Weight*)



Hinge

Hinge

This tells the bone to remain motionless in a chain. It doesn't copy the rotation and scale of the parent. Useful for mechanical rig I would say, as you can animate the rotation of the hinge bone without having to correct it because the parent rotated. (*Hinge*)

Deform

This lets you tell if you want the bone to deform at all. It's like setting weight to 0, except it's faster this way. Useful when using a bone as a target or a controller, i.e. a bone you just want to use to work on other bones.

Mult

To deform geometry you can use vertex group and/or Envelope. The fact that you can mix both ways gets interesting when you can use one to better tweak the other one. Like using envelope everywhere but tweaking a difficult place manually with vertex group. It's gonna be shown more in details in the Skinning section.

Hide

This option lets you hide the bone. You can use it to hide useless bones when you try to see what you're doing or just to get a functional rig for an animator when everything is done, kind of "hiding the useless". For example, when you animate you don't need the entire chain of the leg, just the controllers. This option is valid for both EditMode and PoseMode. It's possible to directly Hide bones in the 3Dview by selecting the bones to hide and do H. You can turn all bone visible with Alt H too.

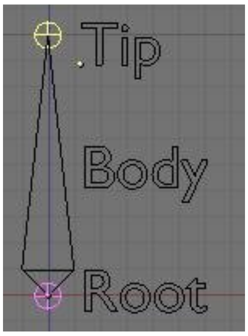
Selecting Bones

Mode: Edit Mode (Armature)

Hotkey: RMB

Menu: Select Select/Deselect All, Select Border Select

Description



Bone parts

Bones in Edit mode consist of the tip (the end pointing to) and the root (the end pointing from), and these can be moved independently. RMB clicking on the tip or the root selects either, or clicking on the bone's body selects the entire bone. Selecting both the tip and root selects the entire bone implicitly.

L selects a chain of connected bones under the mouse pointer

P selects the parent bone of selected bones. This is useful when using a fullscreen 3d−View, so an Outliner view is not required to be able to select the parent bone(s). This also works in pose mode.

Adding and Deleting Bones

Mode: Edit Mode (Armature)

Hotkey: Shift A, E, Shift D

Menu: Armature Extrude, Armature Duplicate

Description

In Edit mode, you can add a new bone at the cursor location via the Add menu (Shift A).

E extrudes a new bone from a selected root or tip. This will create a bone connected to the original one, meaning the Root of the new bone will follow the Tip of the original one. You can also Ctrl LMB to extrude a new bone. It will extrude to where you clicked.

Shift D duplicates selected bones. An entire bone must be selected, not just the root or tip.

X deletes selected bones

Interactively Setting Bone Roll

Mode: Edit Mode (Armature)

Hotkey: Ctrl R

Description

Setting the roll value of bones manually has always been a chore. Sometimes, especially on more complex rigs, it may be necessary to manually assign the roll values. However, adjusting the roll value could only be done for one bone at a time, and it was difficult to gauge what the right amount to adjust it was.

Now there is a new transform mode/tool in EditMode. You can select multiple bones and set their roll values interactively. It acts like rotating the bones around the local y−axes in PoseMode. For better feedback, it is recommended that you turn on *Draw Axes* for the Armature in question.

You can access this tool by pressing Ctrl R

Subdivide

Mode: Edit Mode (Armature)

Hotkey: W

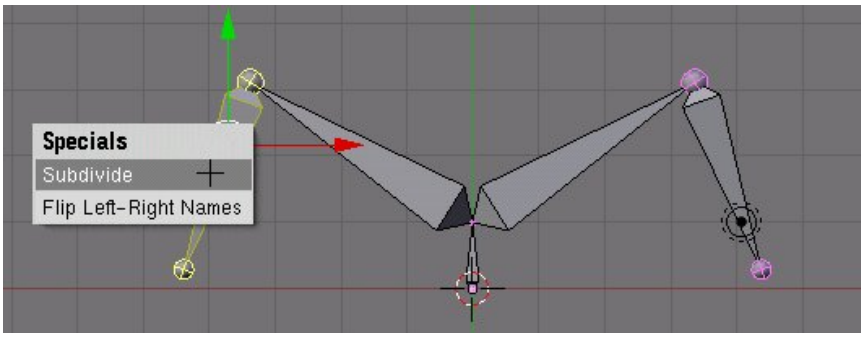
Menu: Armature Subdivide

Description

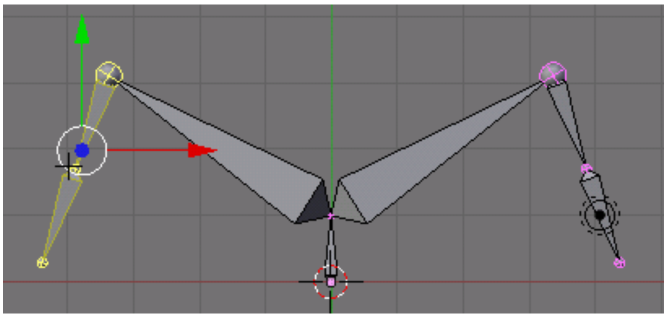
You can subdivide a bone (*Before/After subdivision*) with the W menu. The menu gives you the following options:

- Subdivide − this simply divides the bone into two bones, as shown in the graphic.
- Subdivide Multi − this option gives you a pop−up that asks for the number of cuts. It defaults to the last value used. If you want to change one bone into five, you'll want four cuts. Use the triangles on either side to increment/decrement the field, or click on the field and enter the number you want yourself. Press the OK button when you're done to complete the operation.
- Flip Left−Right Names − Blender will try to help you name bones assuming a right/left symmetry. Sometimes, it gets right and left mixed up, use this option when you know that's going to happen, so you don't have to manually rename the bones.

Flipping Left and Right Bone Names



Before subdivision



After subdivision

X−Axis Mirror Edit

Mode: Edit Mode (Armature)

Panel: Editing Context Armature

Description

X−axis mirroring replicates edits to one side of an Armature, mirrored on the other side. This works many of Blender's armature editing tools, including manipulations such as move, rotate, scale, for extrude and subdivide. It's a clean way to just do half the job.

The axis of mirroring is X so left<−−>right in frontview (NumPad 1) and the center is the center of the armature object.

Flipping Left and Right Bone Names

Mode: Edit Mode (Armature)

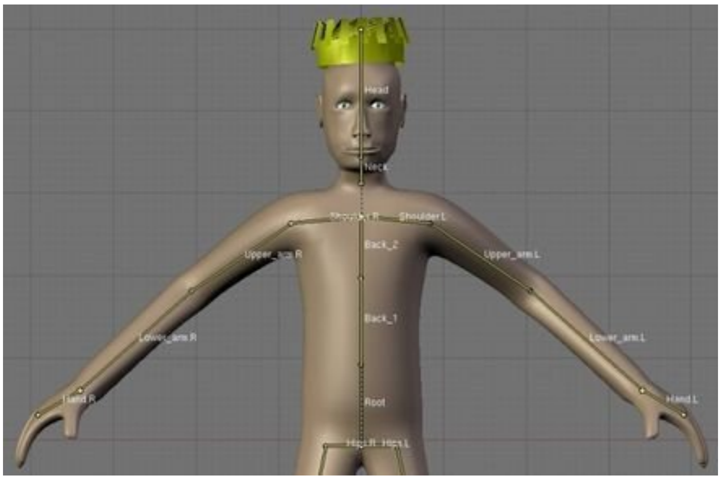
Hotkey: W

Menu: Armature Flip Left & Right Names

Description

Blender can flip left or right markers in bone names. This can be useful if you've constructed half of a symmetrical rig (marked for a left or right side) and duplicated and mirrored it, and want to update the names for the new side. Blender will swap text in bone names according to the following naming conventions.

Naming conventions



An example of left/right bone naming in a simple rig

Naming conventions in Blender are not only useful for you to find the right bone, but also to tell Blender when any two of them are counterparts.

In case your rig can be mirrored in half (i.e. it's bilaterally symmetrical), it's worthwhile to stick to a left−right naming convention. This will enable you to use some tools that will probably save you time and effort.

- First you should give your bones meaningful base−names. Like `leg`, `arm`, `finger`, `back`, `foot`, etc...
- If you have a bone that has a copy on the other side (a pair), like an arm, give them one of the following separators...
    - Left/right separators can be either the 2nd position (`L_calfbone`) or last−but−one (`calfbone.R`)
    - If there is an lower or upper case `L`, `R`, `left` or `right` blender handles the counter part correctly. For a list of separators see below. Pick one and stick to it as close as possible when rigging, it will pay off. For example:

Lefthand	−>	**Right**hand
L Hand.005	−>	**R** Hand
hand**.r**	−>	hand**.l**
Foot**−l**	−>	Foot**−r**
pelvis **LEFT**	−>	pelvis **RIGHT**

- Before Blender handles an armature for mirroring or flipping it first removes the number extension, if it's there (like `.001`)
- You can copy a bone named `bla.L` and flip it over using W >> flip name. Blender will name the copy `bla.L.001` and flipping the name will give you `bla.R`. Extensions such as `.001` are also

preserved.

Possible separators for Left−Right extensions:

- space " "
- dot "`.`"
- minus "`-`"
- underscore "`_`"
- If no 'switch' done yet, it tries if a name starts or ends with `left` or `right`, case insensitive. It replaces this, disregarding separator.

Previous: Manual/Armature Objects Contents *Next:* Manual/Posing Armatures

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.40**

Armature Posing Options

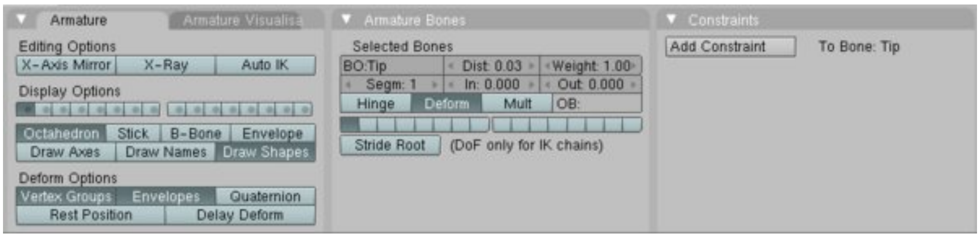
Mode: Edit Mode (Armature)

Panel: Editing Context Armature

Description

Contrary to [EditMode](#), Pose mode isn't a obligatory mode where you can't do anything else. You can be in posemode and still select another object. When you are done building your armature, you can go into [PoseMode](#) to add constraints and start creating actions.

Options



PoseMode panel

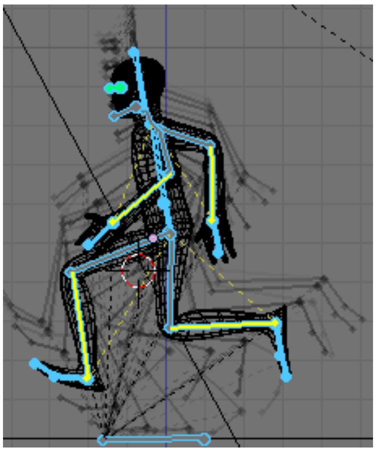
Automatic IK
: Use this feature in the Editbutton (F9) to pose a chain of bones like it was an IK chain. The usefulness is very limited though. It works well only if there is no other IK solver in the chain, and if your chain is isolated from the rest of the rig.

Ghost
: In the Armature Visualisation panel the Ghost option, if set > 0, lets you see the action linked to the armature over time. Also called onion skinning.(*Ghost*)

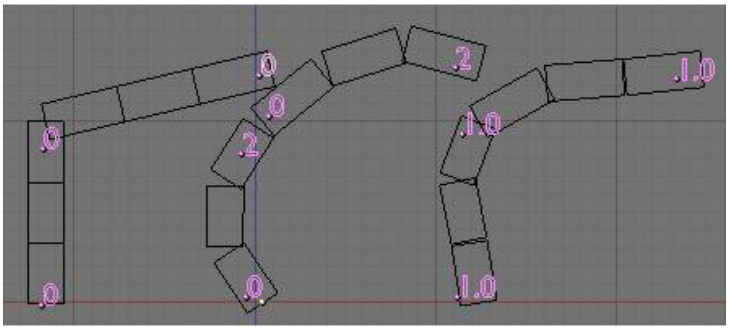
In / Out
: There are two number fields used to tweak the effect of B−Bones. The In:/Out: is used to tell the scale of the virtual handle of the bezier curve. In: is the root of the bone and Out: is the tip. The bigger the value, the bigger the effect of rotation. (*B−Bones In/Out*)

Limit Rotation

If the bone participates in an IK constraint chain, there will be three more sub−panels on the Armature Bones panel, enabling you to Lock (prevent any changes), set a joint stiffness, or to limit the rotation within a degree min/max. Using these controls, you define an envelope (shown visually) as to the 3D space that the tip of the bone can point to, when following or responding to an IK target.



Ghost



B−Bones In/Out

Armature Posing Tools

Mode: Edit Mode (Armature)

Panel: Editing Context Armature

Description

- You can pose your rig using G, S and R. Note that if the bone is part of a chain it can't be moved (except if it's the first of the chain, moving all the chain as they are all children), so you rotate the bone instead.
- You can do Alt S on one or more bones while in Envelope display mode to tweak the envelope size in real time while animating. Useful when for example you move the hand and some part of the character isn't in the influence zone; the result will be that some vertices will stay behind.
- You can do Ctrl C to copy stuff from one bone to a group of bones. The options are location, rotation, scale and constraint. Constraint is very handy when you want to copy a constraint to other bone. The way it work is easy.
- The W menu get some neat options too:
    - Select constraint target: Will select the target of the bone's constraint currently selected.
    - Flip name: Yep you can flip name in PoseMode too.
    - Calculate path/Clear path: This is a visual way to see the action linked to your armature. You can select just some bones and ask Blender to show you the path of the bone.
- You can pose your character and select all bone you want to see included in the action and press I. You can insert a key just for loc, rot or size. Avail will add a key to all available channels in IPO window (all channel you previously added something).
- When you insert a key for your armature, a new action is created and linked to the armature if there was no action before. You can also see the curves of each selected bone of the armature in the IPO window.
- You can parent a bone to an external object by selecting this object then selecting the bone in question so it's active (The armature is in PoseMode so you can select a bone). Do Ctrl P. Then when you move the bone the object will follow. This kind of Hard relationship doesn't allow multiple bone influence like a vertice. It's useful when doing robot rigs as you are just moving objects around.

Previous: Manual/Editing Armatures	Contents	*Next:* Manual/Inverse Kinematics

User Manual: Contents | Guidelines | Blender Version **2.46**

Inverse Kinematics

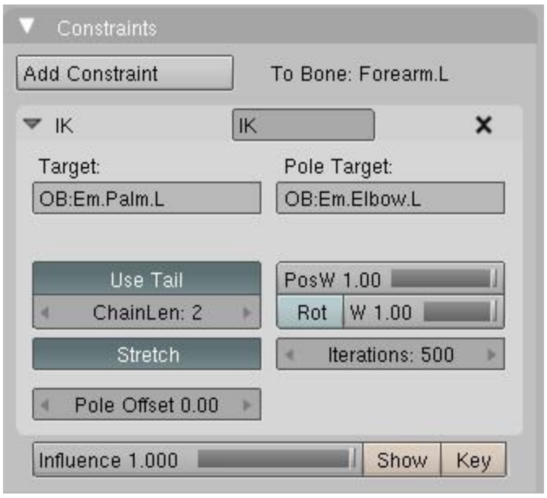
[Much work has been done on the Inverse Kinematics Solver] both to improve its performance and to add new features. Additionally, setting up IK is now easier than ever!

Interface

To enable IK solving, select the tip of the bone chain and press Ctrl I, choosing "To New Empty Object" from the popup menu. A new empty is created, and is set as the target of the IK chain. The bone is colored yellow, and a dashed line indicates how far up the chain of bones the IK solver will work. The new "ChainLen" option of the IK Solver constraint (found in the Object buttons panel) allows control over how far up the chain the IK solver extends, replacing the old "IK" toggle in edit mode. A chain of 0 is **all** bones from that bone back through and connected to it, all the way back to the root bone. This also means that bones in an IK chain no longer have to be connected, but allow an offset from their parent. The other popup menu option, "Without Target", is discussed at the end of this article, under "Mixing IK and FK".

In order to use a bone as an IK target (without creating an empty), select the target bone first, then hold down SHIFT and select the bone at the tip of your IK chain. Pressing Ctrl I and choosing "To Selected Bone" from the popup menu sets the tip bone to use the target bone for its IK target. Note that the target bone cannot be a part of the bone chain that will have IK.

Tree IK



There is now support for "tree IK". IK chains starting in the same bone will automatically be solved together, affecting each other as needed to reach their targets. The "PosW" slider defines the importance of each target, in case not all targets can be reached.

IK target rotation

The "Rot" option of the IK Solver constraint makes the tip of the chain match the rotation of the target. Its importance relative to reaching the position of the target can be controlled with the "RotW" slider.

IK Pole Target

This bone constraint allows you to define a "pole", which is the direction in which a knee, or middle joint in the IK chain, should point track to. [See Release log for more info.] The bone IK Solver constraint for a bone chain has both an IK target (what the end bone points to) as well as a Pole Target, which is the target that the chain points to, or stays "facing" as it bends. This is most useful in knees and elbows, but would also work to allow you to control which way a tail "crinks" when compressing. Parent an empty to the armature, and then use it as a pole target. Animate the location of that empty, which would guide the bone chain's orientation as it bends. For knees, float the empty out in front of the leg (study mocap to see how the leg/hip rotation results in knee orientation during the walk cycle). For elbows, float the empty behind the arm, and study mocap to see how the elbows fly out or stay tucked in for different movements.

What is a Pole Target?:

"*A pole target is a secondary target for a bone with an IK constraint.*
The first target is where the chain of bones is trying to get to, and the second target (pole target), is where the chain bends to to get to this target.
A possible setup is this: A chain of bones (like upperarm−>forearm) has an IK constraint on the last child bone (forearm) and is set to target an IK target (like a replacement target bone for the hand). Then to control the direction the elbow is pointing, one uses another target, the pole target."

−− Quote (slightly modified) of FreakyDude from this thread.

Rotation limits

The behavior of individual bones in an IK chain can be modified. For bones in an IK chain there are a number of options available in pose mode, in the Edit buttons.

By default bones have 3 degrees of freedom (DoF), meaning they can rotate over the X and Z axes, and roll or twist along the Y axis. Each DoF can now be locked, to disable rotation over that particular axis.

As an example, let us consider a human arm. The wrist has 2 DoF. It can bend in any direction, but it cannot twist. The elbow also has 2 DoF: it can twist, and bend in one direction. Finally, the shoulder has 3 DoF. An example of a 1 DoF joint is the knee.

The "stiffness" defines per DoF how eager the bone is to rotate.

Setting rotation range

The range of rotation can also be limited. "Limit X" and "Limit Z" define how far the bone can rotate over the X and Z axes respectively. If both are enabled this defines an ellipsoid region. "Limit Y" defines how much the bone can twist.

There are two important things to remember:

- DoF and rotation limits are defined with respect to the rest position of the bone.
- They only work for bones in IK chains.

Mixing IK and FK

In many cases you only want to use IK to assist in posing characters, without having an IK defining the motion at all times during an animation. For that purpose, two features were added:

- Targetless IK

When an IK chain has no target defined, it can still be used for posing. Unlike normal IK, you must set keys on all of the bones in the chain if you want to retain the pose. Bones using this type of IK are drawn in a orangish color.

- Automatic IK

This option, in the Editing Buttons, "Armature" Panel, automatically assigns a temporary IK chain to any translated bone, giving the same effect as if the selected bone had been assigned Targetless IK. This chain then only propagates over the connected Bones of the grabbed one.

Previous: Manual/Posing Armatures	Contents	*Next:* Manual/Mesh Skin Weighting

User Manual: Contents | Guidelines | Blender Version **2.4x**

Skinning

V.: 2.31

Once the Armature − the 'character skeleton' − is ready it is necessary to parent the character 'skin' to it. Skinning is a technique for creating smooth mesh deformations with an armature. Essentially the skinning is the relationship between the vertices in a mesh and the bones of an armature, and how the transformations of each bone will affect the position of the mesh vertices. When making a child of an armature, several options are presented:

- Parent to Bone

  In this case, a popup menu appears allowing you to choose which bone should be the parent of the child(ren) objects. This is great for robots, whose body parts are *separate* meshes which are not expected to bend and deform when moving.

- Parent to Armature

  Choosing this option will deform the child(ren) mesh(es) according to their vertex groups. If the child meshes don't have any vertex groups, they will be subject to automatic skinning. Indeed a second menu appears, asking:

  - ◊ Don't create groups − does nothing else, automatic skinning is used;
  - ◊ Name Groups − creates empty vertex groups whose names matches the bone names, but no vertices are assigned to them;
  - ◊ Create from closest bone − you want to create *and* populate automatically vertex groups.
  - ◊ Create from bone heat − creates vertex groups and assigns weights to each vertex using the 'bone heat' algorithm. This sometimes 'fails to find a solution'; if this happens, in edit mode use P > All loose parts, which will split the mesh into multiple meshes, and then attach the armature to each mesh.

- Parent to Armature Object

  Choosing this option will cause the child(ren) to consider the armature to be an Empty for all intents and purposes.

If you are going for character animation then most of the times you will parent your character to the Armature using the "Armature" Option. You are strongly advised to use the Name Groups option. This will provide you with the groups already created, saving the tedious operations of creating an naming them, and possibly avoiding typing errors. The Create from closest bone feature is currently under heavy development. It will use the "Bone types" which can be defined via the menu right of the IK Tog Buttons (*EditButtons for an Armature*) for optimal result. Currently only the Skinnable and Unskinnable options are working. The first option makes Vertex Group be created (and populated, if this is asked for) for the given bone, the second option causes that bone to be ignored in the skinning process.

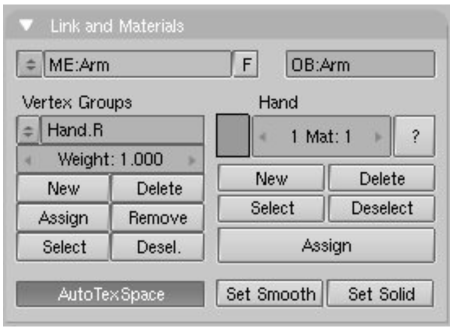
Automatic Vertex Assignment

The current vertex assignment algorithm creates non−optimal vertex groups, hence it is highly recommended to check each group, one by one.

If a mesh does not have any vertex groups, and it is made the child of an armature, Blender will attempt to calculate deformation information on the fly. This is very slow and is not recommended. It is advisable to create and use vertex groups instead.

Weight and Dist

The *Weight* and *Dist* settings next to the *IK* are only used by the automatic skinning which is a deprecated feature because it requires lot of CPU, produces slow downs and worse result than other methods.



Vertex Groups.

Vertex groups are necessary to define which bones deform which vertices. A vertex can be a member of several groups, in which case its deformation will be a weighted average of the deformations of the bones it is assigned to. In this way it is possible to create smooth joints.

To add a new vertex group to a mesh, you must be in Edit Mode. Create a new vertex group by clicking on the New button in the mesh's Edit Buttons Link and Materials Panel, Vertex Groups group.

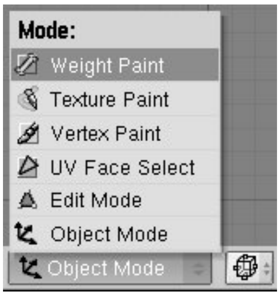
A vertex group can be subsequently deleted by clicking on the Delete button. Change the active group by choosing one from the pull−down group menu. Vertex groups must have the *same* names as the bones that will manipulate them. Both spelling and capitalization matter. This is why automatic name creation is so useful!

Rename a vertex group by SHIFT−LMB on the name button and typing a new name. Note that vertex group names must be unique within a given mesh.

Vertices can be assigned to the active group by selecting them and clicking the Assign button. Depending on the setting of the Weight button, the vertices will receive more or less influence from the bone. This weighting is only important for vertices that are members of more than one bone. The weight setting is not an absolute value; rather it is a relative one. For each vertex, the system calculates the sum of the weights of all of the bones that affect the vertex. The transformations of each bone are then divided by this amount meaning that each vertex always receives exactly 100% deformation.

Assigning 0 weight to a vertex will effectively remove it from the active group. To remove vertices from the current group select them and click the Remove button. Pressing the Select button will add the vertices assigned to the current group to the selection set. Pressing the Deselect button will remove the vertices assigned to the current group from the selection set. This is handy to check which vertices are in which group.

Weight Painting



Weight Paint Button.

Weight painting is an alternate technique for assigning weights to vertices in vertex groups. The user can "paint" weights onto the model and see the results in real−time. This makes smooth joints easier to achieve. To activate weight−painting mode, select a mesh with vertex groups and click on the weight paint icon (*Weight Paint Button.*).

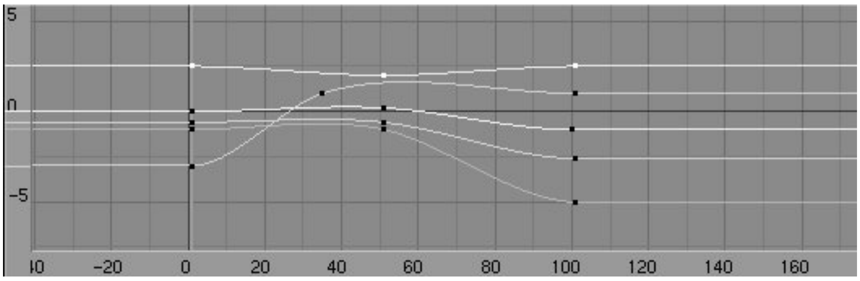
The active mesh will be displayed in Weight−Colour mode. In this mode dark blue represents areas with no weight from the current group and red represent areas with full weight. Only one group can be visualized at a time. Changing the active vertex group in the Edit Buttons will change the weight painting display.

Weights are painted onto the mesh using techniques similar to those used for vertex painting, with a few exceptions. The "colour" is the weight value specified in the mesh's Edit Buttons. The opacity slider in the vertex paint Buttons is used to modulate the weight. To erase weight from vertices, set the weight to "0" and start painting.

Baking Actions

If you have an animation that involves constraints and you would like to use it in the game engine (which does not evaluate constraints, and is not covered in this Book), you can bake the Action by pressing the BAKE button in the Action Window ToolBar. This will create a new Action in which every frame is a KeyFrame. This Action can be played in the game engine and should display correctly with all constraints removed. For best results, make sure that all constraint targets are located within the same armature.

You can actually see the Action Ipo associated to a bone in the Ipo Window instead of in the Action Window if you switch to an Ipo Window (*Action Ipo*). The Action Ipo is a special Ipo type that is only applicable to bones. Instead of using Euler angles to encode rotation, Action Ipos use quaternions, which provide better interpolation between Poses.



Action Ipo.

Quaternions use a four−component vector. It is generally difficult and unintuitive to describe the relationships of these quaternion channels to the resulting orientation, but it is often not necessary. It is best to generate quaternion KeyFrames by manipulating the bones directly, only editing the specific curves to adjust lead−in and lead−out transitions.

Context menu

Weight paint mode also has a 'Paint' context menu with the following options:

- Apply bone envelopes to vertex groups − calculates vertex groups and weights from the bone envelopes
- Apply bone heat weights to vertex groups − similar to the option when initially parenting (see above), uses the 'bone heat' algorithm to create vertex groups and assign weights to vertexes
- Various scripts

Previous: Manual/Inverse Kinematics | Contents | *Next:* Manual/The Action Editor

User Manual: Contents | Guidelines | Blender Version **2.43**

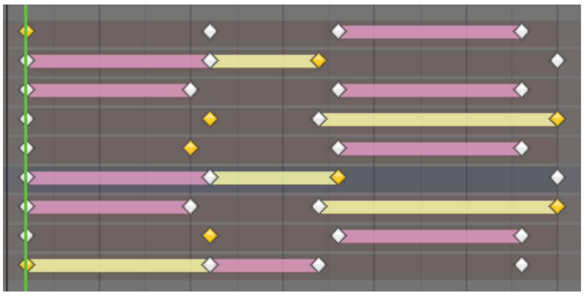
The Action Editor provides control over the keys set for bones in an armature, as well as for Shape Keys.

Long Keyframes

When animating, it is often useful to be able to visually see where the 'pauses' are between keyframes. Long keyframes do this − linking two keyframes in the same channel together.

Long keyframes are only drawn when the two keyframes have the exact same values. This has to happen for every ipo−curve represented by the keyframes shown for a long keyframe to be drawn.

There are two new theme colours for the action editor. They are for the selected and deselected colours of the long keyframes (currently defaulted to be the same as the NLA strip selection colours).



Long Keyframes

Shape Key Sliders

Due to a few internal cleanups of code, it is now possible to save the visibility status of Shape Key sliders. Also, there are no more problems with having multiple Action Editors open, and all displaying Shape−Key actions.

Action Window

An Action is made of one or more Action channels. Each channel corresponds to one of the bones in the armature, and each channel has an Action IPO associated with it. The Action Window provides a means to visualize and Edit all of the IPOs associated with the Action together.

Selection Tools

Selection tools in the Action Editor have always been limited to only a few basic ones. This release, several new options have been added.

Invert Keys
: Selects the keyframes that were not selected, and deselects the keyframes that were selected.

Column Select â†’ On Selected Keys

Selects all the keyframes that occur on the same frame as selected keyframes. Use the hotkey K.

Column Select â†’ On Selected Markers
Selects all the keyframes that occur on the same frame as selected markers. Use the hotkey Shift K.

Column Select â†’ Between Selected Markers
Selects all the keyframes that occur between and on the first and last selected markers. Use the hotkey Ctrl K.

Mirror Tools

Like in the Ipo Editor, it is now possible to 'mirror' keyframes over a line. This makes it easier to reverse an action. Four options are currently available:

Mirror Over Vertical Axis
Mirrors selected keyframes using frame == 0 as the mirror−line.

Mirror Over Current Frame
Mirror selected keyframes using current frame as the mirror−line.

Mirror Over Horizontal Axis
Mirrors selected keyframes using value == 0 as the mirror−line.

Mirror Over Selected Marker
Mirrors selected keyframes using the frame of the first (chronologically) selected marker as the mirror−line.

Hotkey for Mirror tools is Shift M.

Snap Tools

Shift−S Snapping

So far, snapping tools have been restricted to 'Snap to Nearest Frame' and only for action channels. Obviously, this is too limited. Now, you can also snap keyframes to the current frame:

- Snap To Nearest Frame
- Snap To Current Frame
- Snap To Nearest Marker

Hotkey for Snap tools is Shift S

It is also worth mentioning, that these now work correctly with actions scaled in the NLA editor. Previously, there could be unpredictable results as there was no correction applied.

Auto−Snapping During Transforms

Many people have requested such an option, as it reduces strain on fingers. There's a new selection−box on the header of the action editor, which sets the mode of auto−snapping for transforms. By default auto−snapping is off.

There are 3 modes of auto−snap:

- Off − transforms per normal
- Frame Step − grid−step transform (may have errors with scaled actions)
- Nearest Frame − true snap−to−frame (takes into account nla−scaling)

These translate to the following hotkeys when transforming:

- Off − no keys press/held (as it's always been)
- Frame Step − Ctrl (as it's always been)
- Nearest Frame − Shift (replaces old shift−key behaviour which was not useful)

Channel 'Protecting'

Action Channels and Constraint Channels can be 'protected' (aka locked). That means that the keyframes for that channel cannot be moved (by any transforms or operations which modify the times the keyframes occur at), be duplicated or destroyed, or have their handle type changed.

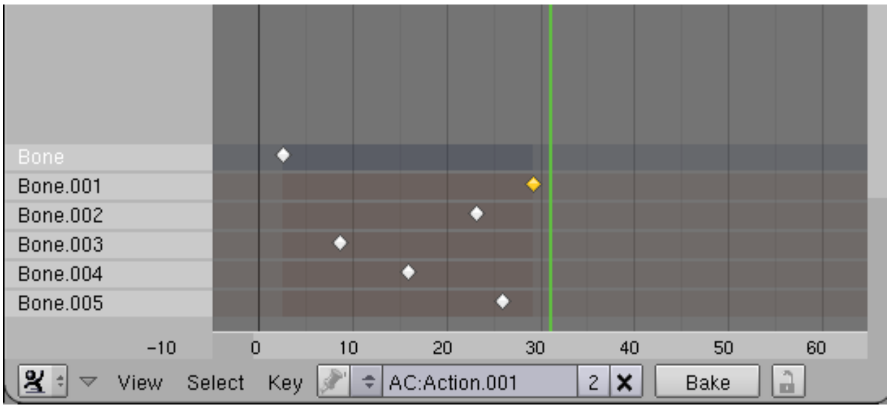
This is useful for protecting parts of animation that has already been finalised, while other parts are worked on.

There is a lock icon on the right−hand side of the channel names. This icon serves two purposes − as an indicator or whether the channel is locked, and also as the way to turning locking on/off for the channel in question.

Channel locks also have effect when the action's keys are displayed in the NLA editor. However, they don't currently have any connection with the ipo editor and ipo curves, so it is still possible to insert keyframes and modify keyframes from the ipo editor directly.

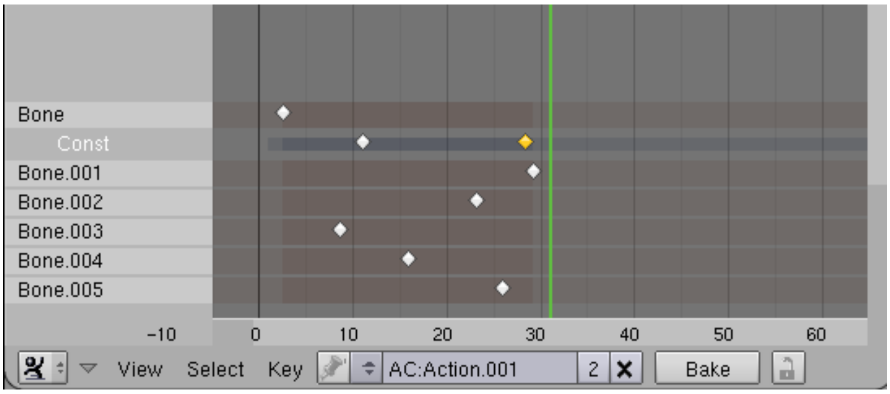
Tip

You can activate the Action Window with SHIFT−F12



The Action Window

For every key set in a given Action IPO, a marker will be displayed at the appropriate frame in the Action Window. This is similar to the "Key" mode in the IPO Window. For Action channels with constraint IPOs, there will be one or more additional constraint channels beneath each Action channel. These channels can be selected independently of their owner channels.



Action Window with a Constraint

A block of Action keys can be selected by either RMB on them or by using the boundary select tool B. Selected keys are highlighted in yellow. Once selected, the keys can be moved by pressing G and moving the mouse. Holding CTRL will lock the movement to whole−frame intervals. LMB will finalize the new location of the keys, while ESC cancels the Action and returns to previous state.

Using ALT RMB in the workspace of the Action Editor will select all keyframe markers on that side of the current frame marker.

A block of Action keys can also be scaled horizontally (effectively speeding−up or slowing−down the Action) by selecting number of keys and pressing S. Moving the mouse horizontally will scale the block. LMB will finalize the operation.

Delete one or more selected Action keys by pressing X when the mouse cursor is over the KeyFrame area of the Action Window.

A block of Action keys can be duplicated and moved within the same Action by selecting the desired keys and pressing Shift D. This will immediately enter grab mode so that the new block of keys can be moved. Subsequently LMB will finalize the location of the new keys. ESC will exit grab, but won't remove duplicates.

You can also delete one or more entire Action or constraint channels (and all associated keys) by selecting the

channels in the left−most portion of the Action Window (the selected channels will be highlighted in blue, and the names will become white). With the mouse still over the left−hand portion of the window, press X and confirm the deletion. Note that deleting an Action channel that contains constraint channels will delete those constraint channels as well.

For more detail see the Action Editor Reference

Previous: Manual/Mesh_Skin_Weighting	Contents	*Next:* Manual/Non_Linear_Animation

User Manual: Contents | Guidelines | Blender Version **2.41**

NLA Editor

Mode: Object Mode / Pose Mode (Armature)

Hotkey: Shift Ctrl F12

Description

You use the NLA Editor to mix actions and IPOs of objects to make a composite animation. Both Object Ipos and Actions can be blended. Every object may have only one composite animation. You can think of it this way: the Action editor says WHAT the bones are supposed to do, and the NLA Editor says WHEN they are supposed to do it. In addition, you can position the object and key its location using the NLA while in the NLA Editor, in order to say WHERE the action is supposed to occur.

The NLA Editor works in two different modes. In the "Action" (single) mode only the current active action is played. To play combined action use the "NLA" (combined) mode.

In the NLA Editor you see a representation of all Objects with either Actions or Ipos. On a per Object basis, you can add "Action Strips" with Shift A; you can add as many Actions as you like, allowing you to mix different Actions to non−destructively edit timing and relationships. Each Object can have only one active Action, which is displayed in the Action Editor. This active Action is the one which will receive any new keys you insert, and whose keys you can directly edit.

When strips are inserted, they start at the current frame. When a strip is added, it becomes the active strip. The current active strip is always drawn in yellow. Clicking on a strip itself or on the left−hand bars will make the strip active. Using the sript properties panel, you can control how long the action takes to execute, or what portion of an action is to play.

You add new Ipo−Keys to the active Action, you can directly edit these Keys (delete, move). The objects you select in the NLA window are selected also in the 3D window. Be careful! Deleting keys in the NLA also deletes them from the Action, and from the underlying IPO curves as well.

Ipo's for objects − and also for armature objects − can be converted to Actions in the Ipo window with Shift C (or from the Strip Menu (Convert Action to NLA strip)

De−Link Actions before starting:

Before entering the NLA, ensure that any object you want to control using the NLA is de−linked from its action. Otherwise, that action will want to play on its own. To de−link, in the Action Editor, be sure Fake User is enabled and press the X button on the header. The Action will remain in memory (for you to call in using the NLA) but will not force the object to perform it. Note also that Actions designated for use in the NLA should start at frame 1, for convenience.

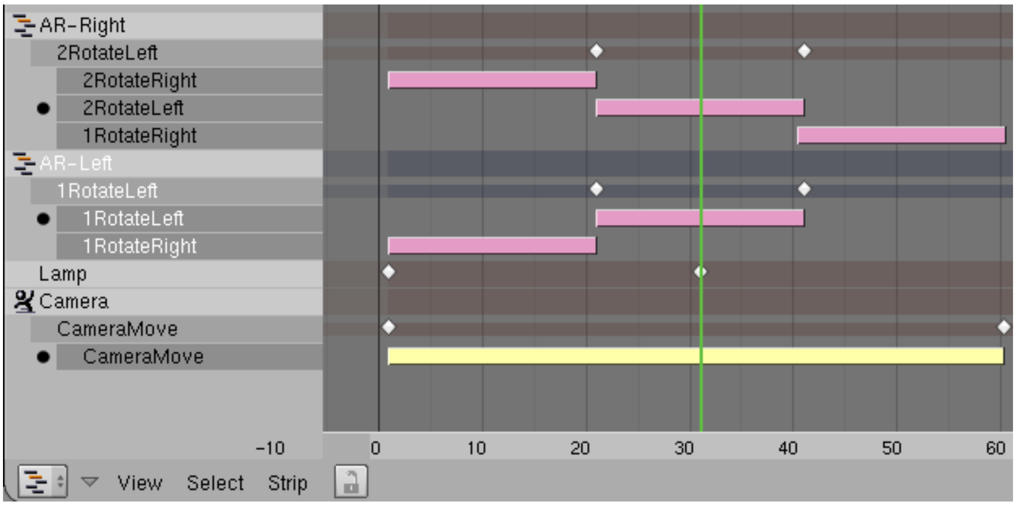


Image 9: Representation of different objects, actions and IPOs in the NLA Editor window.

1) An object is only shown in the NLA editor window if it has Ipos, an action, or NLA strips. The top−most horizontal bar for an individual object shows the Object's name to the left, and any Object Ipo "keyblocks" (the diamonds that represent keyframes) on the right. (*Image 9* − the "Lamp" has only object Ipo's, and no Actions).

2) The button to the immediate left of the Object name toggles Blender between evaluating the entire NLA and displaying the results in the 3D window, or only evaluating the active Action (*Image 9* − "Camera" is in "Action" mode, while "AR−right" and "AR−left" are in "NLA" mode). Note that this icon is only displayed when NLA strips are present for the object.

3) Immediately below the first Object bar, the active Action is shown, and the "keyblocks" for the Action's Ipo's.

4) Finally, the "Action Strips" are drawn. The active Action Strip is indicated with a black dot icon. Clicking on a Strip itself or on the left−hand bars will set a Strip to being the active Action.

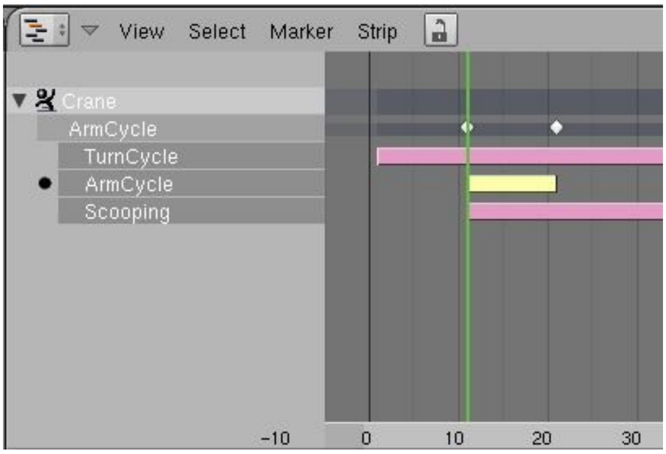
NLA Strip Options

Mode: Object Mode / Pose Mode (Armature)

Hotkey: A

Menu: NLA Editor Strip Add Blank Action or Add Action

Description



Strips are added and shown in an indented fashion. when you Add Action Strip, a popup window allows you to choose from available Actions defined. Add Blank Action creates a new action, by default named "ObAction". To rename it, use the Action Window. The name of the overall action is the name of the armature or object. Below that is an indent strip that shows the name of the currently selected strip and all of the ipo keys for that strip. Indented beneath that are all the loaded action strips.

The example image shows an NLA window for armature Crane in Action mode. The RMB selected strip is ArmCycle, and it has two keys. The white diamond at the current frame (green line) shows that there is a keyed position there. The orange diamond at frame 21 is the currently selected frame (it is ready to be Grabbed and moved, or X deleted).

There are two modes for the editor: Action mode and NLA mode. The mode is shown by the icon at the head of the strip: "drowning man" for Action, "mini−strips" for NLA mode. In Action mode, only the action for a single action strip is animated as you scroll through your animation frames. In NLA mode, all strips are animated as you scroll through your animation frames. When you switch to NLA mode, the armature's bones are all recomputed based on keyed IK and FK positions. Any non−keyed bones will revert to their edit position. So, it is a good idea to key all bones (especially IK drivers) at the start of the animation in the default pose.

Recall that to create actions for armatures, you must Add New Action, then Add New IPO strip, or just key the action in the Action window (and an IPO will be added automatically). Also recall that an action can contain motion and IPO curves for many bones. Therefore, it is possible to load two action strips that have two (possibly conflicting) IPO curves for the same bone. For that reason, you have to use the strip's properties to tell Blender which IPO should take precedence, or if it should blend the two actions together. These and other properties are described below.

The selected strip is shown in yellow and has a big dot next to its indent. All keys for that action are shown as diamonds in the indent strip. The window scrolls vertically with LMB −drag the thumb in the right window margin. It scrolls horizontally and vertically via MMB −drag.

Strips are added in order, the last one at the bottom of the list. To move or re−arrange the strips, sue Strip −> Move or the hotkey Ctrl PgUp or Ctrl PgDn.

Properties

Mode: Object Mode / Pose Mode (Armature)

Hotkey: N

Menu: NLA Editor Strip Strip Properties...

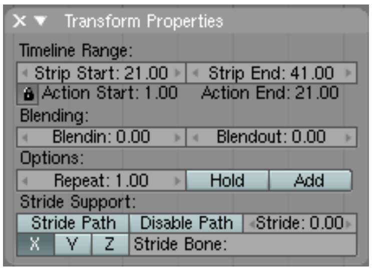


Image 11: The Transform Properties panel for a strip.

The strip properties control how the generic action is applied to the animation. When working on individual actions in Action mode (the "Drowning Man" is shown), the full action is animated in the 3D Window, not just the action between the Action Start and Action End frames in the properties panel. When you switch to NLA mode, only the action range and repeats, along with the rest of the properties, are used to compute the actual animation. These properties include:

Timeline Range
: The first and last frame of the Strip in the timeline. The Strip length is independent from the length of the Action, but the Action will "stretch" to fill the Strip length. This feature allows you to have a generic walk cycle action that spans 40 frames for example. To make your character walk fast, you could enter a 30−frame range; to make him walk slower enter a 50−frame range.

Locked Strip Length
: By setting the strips to Locked, NLA will always keep strip length up−to−date with your Action edits so that all keys are included.
  - ◊ Locked mode is the new default, as it is far superior to the old behavior, but older files will have to be updated manually for this feature to work.
  - ◊ If you release this lock, you can choose only a part of the action to be included in the strip.
  - ◊ Action End is not allowed to be less than Action Start, so you cannot reverse an action in the NLA editor.

Blending
: The number of frames of transition to generate between this Action and the one before it in the Action strip list.

Repeat
: The number of times the Action range should repeat.
  - ◊ This parameter is ignored if Stride Path (in the Stride Support settings) is enabled.

Hold
: If this is enabled, the last frame of the Action will be displayed forever, unless it is overridden by another Action. Otherwise the armature will revert to its rest position.

Add
: Specifies that the transformations in this strip should add to any existing animation data, instead of overwriting it.

Stride Path
: Repeats the strip along a path (for example, to make a walk cycle)

Disable Path
: Temporarily disables the path movement so you can see your action repeating on the spot

Stride
: The distance in Blender Units that the strip should be repeated over (the length of the stride)

X/Y/Z
: The axis of the stride bone to move the armature along

Stride Bone
: The name of the stride bone. See tutorial Stride Support

NLA Strip Tools

Mode: All Modes

Menu: NLA Editor Strip

Description

Edit the properties of the Strips and Keys via the Strip menu:

Options

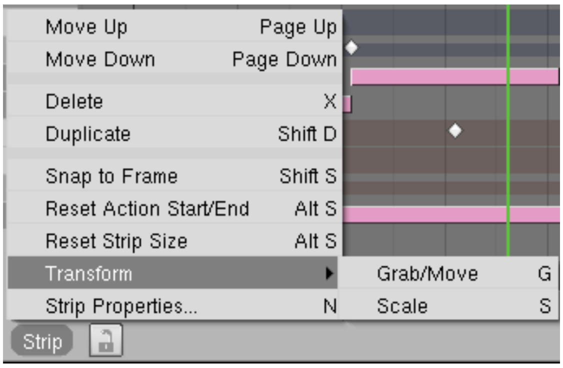


Image 10: The Strip menu.

Move Up/Move Down
: Page Up or Page Down to change the order of the strips. Strips are evaluated top to bottom. Channels specified in strips later in the list override channels specified in earlier strips.

- Delete

X Deletes a Strip or a Key. Deleting a Strip affects only the NLA editor. However, if you delete a key the position is ***permanently*** gone.

Duplicate
: Shift D Copies a Strip or an Ipo.

Snap to Frame
: Shift S Snaps start and end of the selected Ipo Strip to frames.

Reset Strip Size
: Alt S Resets the size of the Strip in NLA to match the chosen range of frames from Action.

Reset Action Start/End
: Alt S Resets the chosen range of Action frames to include all frames in the Action (this is only necessary if you are not using the new "Locked Strip Length" feature, and is useful for fixing older files or if you have "lost your place" when adding or subtracting frames from Actions).

Grab/Move
: G Moves the strip horizontally. With Ctrl you move by frames.

Scale
: S The base "point" for scaling is the current frame. By setting the frame you can select whether the end, the beginning, or a point in the middle of the strip shall keep its position.

Previous: Manual/The Action Editor | Contents | *Next:* Manual/Constraints/Introduction

User Manual: Contents | Guidelines | Blender Version **2.44**

Constraints are object features that define spatial relationships between objects, and are the standard method for controlling characters among all 3D animation packages that still implement a more traditional approach to digital character animation. In Blender, constraints can be associated to any type of object or bone object, but not all constraints work with bone objects, and not all constraints work with normal world objects.

Interface

The interface that is used for modifiers and constraints is described here:

- Constraint Stack

Constraints

Mode: Any Mode

Panel: Object Context Constraints

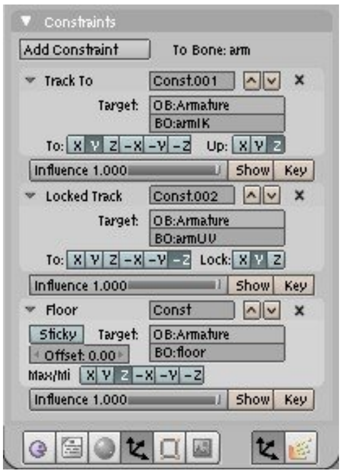
Hotkey: F7 (Panel)

- **Child Of** − Allows a selective application of the effects of parenting to another object.
- **Transformation** −
- **Copy Location** − Copy the position an object based on some other object's position, so that objects move together.
- **Copy Rotation** − Copy the rotation of another object so they dance together.
- **Copy Scale** − Copy the scale of another object.
- **Limit Location** − Object's location is limited to given range.
- **Limit Rotation** − Object's rotation is limited to given range.
- **Limit Scale** − Object's scale is limited to given range.
- **Track To** − Object is tracked to given target object.
- **Floor** −
- **Locked Track** − Object turns to follow a path.
- **Follow Path** − Object moves along a path.
- **Clamp To** −
- **Stretch To** −
- **Rigid Body Joint** −
- **IK Solver** − Bone chain moves to follow another object.
- **Action** − Object executes action based on driving object.
- **Script** − Custom Python script is used as an constraint.
- **Null** −

See BSoD/Introduction_to_Rigging/Constraints_and_Axis_Locks for descriptions of most available constraints.

Previous: Manual/Hooks	Contents	*Next:* Manual/Constraints/Constraint Stack

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.44**



Constraints are accessed via the object buttons (F7) in the Constraints panel. After you press the Add Constraint button and select the desired constraint type from the menu, a constraint UI is added to the panel. The constraint will be linked to the active object or the active selected bone, indicated by the label To Object: or To Bone: on the right of the Add Constraint menu.

Constraints are evaluated in a specific order: from top to bottom of the constraint stack. This order can be viewed and changed inside the Constraints panel. Each constraint has a pair of arrow buttons on the right of the constraint name in its top right corner, which can be used to move the constraint up or down the constraint stack. The little cross on the right of these buttons can be used to delete the constraint from the stack.

NOTE: The name field of a newly added constraint will appear red, which indicates that the constraint is not functional. In the case of a newly added constraint this is because the constraint does not yet have a target specified. All constraints require a target object (a normal world object or an armature bone). You should enter the name of the desired target object in the Target: OB field. When you want an armature bone as target, enter the name of the armature object as target. A new field with BO: will appear, where you can place the name of the target bone.

The name field of a constraint will also turn red if the settings for the constraint are invalid or if the constraint conflicts with another constraint. For example: A [Track To](#) constraint of which the To and Up vector are set to Z.

Influence

The **influence** of a constraint on the actual location/rotation/size of the object can be defined with the Influence value slider. This can be linked an Ipo Curve, which can be evoked in the Ipo Curve Editor with the button Show on the right of the Influence value slider. The Key button beside it can be used to add a key to the Ipo Curve.

You can key any constraint, and an object can have multiple constraints keyed to exert a different influence at different times.

Multiple Influence Example

Suppose that you want a camera to follow one path for awhile and then drift off and follow another path. You apply a *follow path* constraint to a camera for one circle/curve/path (make sure you have curve path enabled and enter its name), then open up an IPO curve window. In the buttons window (constraint panel) you can

either hit I and insert an Influence value by sliding the slider (to vary the force strength or force fall off) or LMB click the "key" button at the bottom of the constraint panel. Once you have inserted the initial key, up arrow or change to the frame where you want the first influence to start falling off. Go to the IPO window and and Ctrl LMB to insert a key. Go forward a few frames, where you want its influence to be zero, and insert another key, and edit it so that its Y value is zero. At that frame, the constraint in the stack will not influence the camera motion at all. Now insert a *second* Follow Path constraint, and enter the name of the *second* path. Now position back a few frames and insert a key to where you want its influence to start (probably where the first constraint starts dropping off). Edit this second influence curve to come up from zero to one over time. You can imagine that the IPO curves for each circle should cross over each other so that when one is at full influence (max 1.0) the other is at zero etc. You might have to play with the curves a bit to get the transition smooth.

Constraints on Bones

Concerning bones with constraints: the color of the bones in the 3D Window indicate what type of constraint is used:

- **Grey**: No constraint.
- **Yellow**: A bone with an IK constraint.
- **Orange**: A bone with an IK constraint but no target.
- **Green**: A bone with any other kind of constraint.
- **Blue**: A bone that is animated with keyframes.
- **Purple**: The Stride Root.

Previous: Manual/Constraints/Introduction	Contents	*Next:* Manual/Constraints/Child Of

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.46**

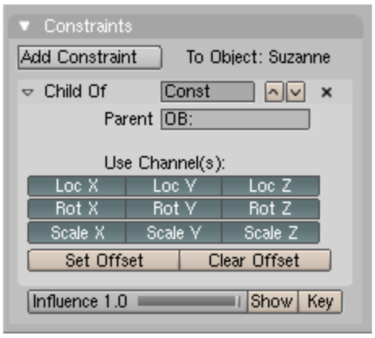
Child Of Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



This certain constraint allows for animatable and or multiple parenting relationships. Now on to the buttons!

Options

Lock X, Y, and Z
: Each of these options will make the parent affect the location according to that axis.

Rot X, Y, and Z
: Makes the parent change the specified axis's rotation.

Scale X, Y, and Z
: Parent affects the scaling on the selected axis.

Set Offset
: Restores the offset (distance) between the objects before the parent relationship.

Clear Offset
: This button moves the object back to where the parent sets it.

Show
: Shows the "Constraints" IPO in the IPO Editor. It will also add a channel

for the IPO if there is not already one.

Key
: This little button sets a keyframe for your work all in a simple click.

Tips

When creating a new parent relationship using this constraint, it is usually necessary to click the 'Set Offset' button after assigning the parent. This cancels out any unwanted transform from the parent, so that the owner returns to the position + orientation it was in before the constraint was applied. Note that you should apply 'Set Offset' with all other constraints disabled for a particular child−of constraint.

There are also toggles to enable/disable individual transform channels from the parent affecting the owner. In practice, it is usually best to leave them alone, or disable all of the toggles for a particular transform.

Example

Previous: Manual/Constraints/Constraint Stack	Contents	*Next:* Manual/Constraints/Transformation

User Manual: Contents | Guidelines | Blender Version **2.46**

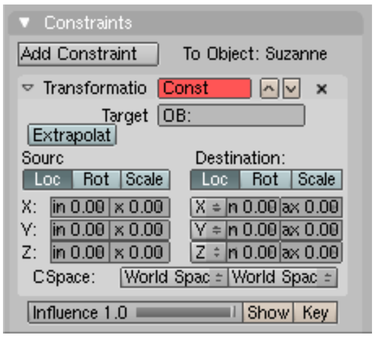
Transformation Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Options

Extrapolation
: description here

Source
: description here

Destination
: description here

CSpace
: description here

Example

Previous: Manual/Constraints/Child Of Contents *Next:* Manual/Constraints/Copy Location

User Manual: Contents | Guidelines | Blender Version **2.44**

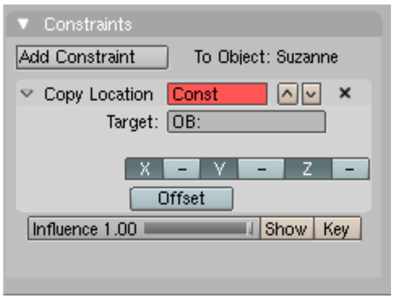
Copy Location Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Copy Location forces the object to have the same location as its target. When this constraint is used on a bone and another bone is the target, a new button−−labeled **Local**−−appears next to the **X Y Z** buttons. Using the local button causes the local translation values of the target to be given to the constrained object. In rest position, all bones are at the local location of (0,0,0).

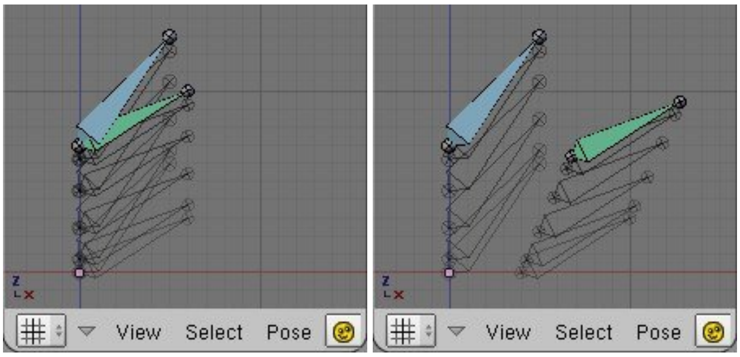
Options

Target
: Object of which location to copy.

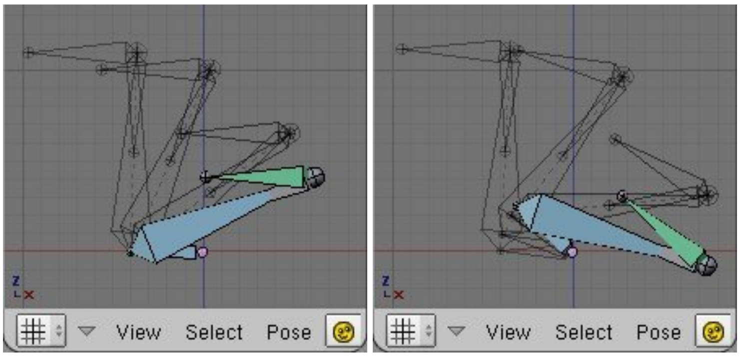
X Y Z
: Axis to constraint. Use − to invert it.

Head/Tail − With Bone targets only
: A number from 0.0 to 1.0 that represents the place on the target bone to use for the Copy (0.0 = the bone's root; 1.0 = the bone's head)

Example



Left: Using global space. This is the default behavior; it's what happens when the local button is not activated.
Right: Using local space, with local button activated.



In these last two images, the constrained bone (shown in green) is actually a child of the root bone (the bone at the beginning of the chain). This demo shows possible uses for the location constraint in a rig. Note that the green bone still inherits rotation from the root because the root is its parent. This is by design though; the green bone could be the child of any of these bones, or none of them.

Previous: Manual/Constraints/Transformation	Contents	*Next:* Manual/Constraints/Copy Rotation

User Manual: Contents | Guidelines | Blender Version **2.44**

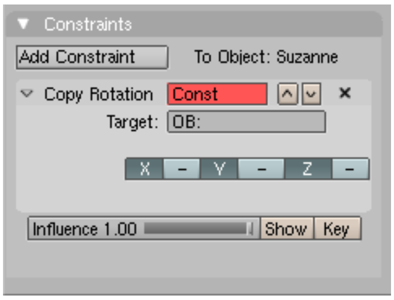
Copy Rotation Constraint

Mode: Object Mode and Pose Mode

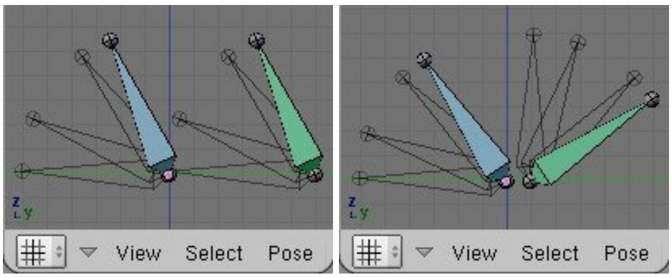
Panel: Object Context Constraints

Hotkey: F7

Description



Copy rotation causes one object to match the rotation of a target object. For bones, a local option will appear, allowing you to use local space. Imagine you have a bone pointing up and a bone pointing down. If you use local space, each can point different directions, but when the target bone moves to its left, the affected bone will move it to *its* left.



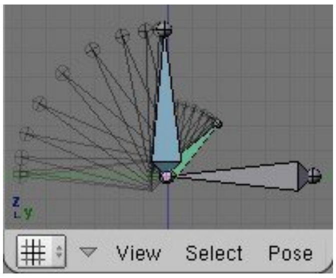
Left: Using global space. Right: Using local space.

Options

Target
: Object of which rotation to copy.

X Y Z
: Axis to constraint. Use the button with the minus sign (−) to invert it.

Example



Here is one good use of the rotation constraint and local space. By setting the influence value to 0.5, the small bone will rotate half as far as the target. This is useful for character joints.

Previous: Manual/Constraints/Copy Location	Contents	*Next:* Manual/Constraints/Copy Scale

User Manual: Contents | Guidelines | Blender Version **2.44**

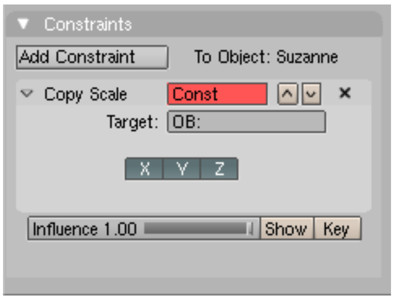
Copy Scale Constraint

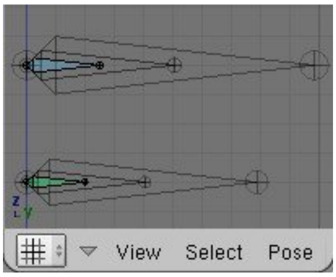
Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description





Copy Scale forces the affected object to have the same size as the target. All bones have a (1,1,1) size in rest position. We can draw a bone that is 10 million times longer than the bone right next to it, but in pose mode, they both have a starting size of (1,1,1). You should keep this in mind if you're going to be using the copy scale constraint.

Options

Target
: Object of which scale to copy.

X Y Z
: Axis to constraint.

Example

Previous: Manual/Constraints/Copy Rotation	Contents	*Next:* Manual/Constraints/Limit Location

User Manual: Contents | Guidelines | Blender Version **2.44**

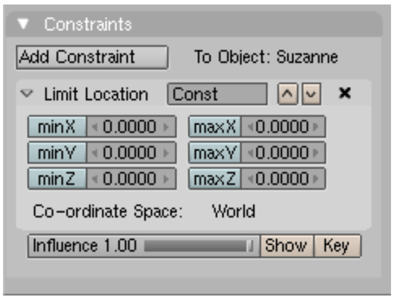
Limit Location Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



An Object can be moved around the scene in the x, y and z coordinates. This constraint places a limit on the location of an object. A limit can be specified on the upper and lower bounds of each coordinate direction. Separate upper and lower limits can be applied for each of the three spatial coordinates (x, y and z). It is interesting to note that even though the limit constrains the visual and rendered location of the object, the object's data block still allows the object to have coordinates outside the minimum and maximum ranges. This can be seen in the Transform Properties N. When an object is grabbed and attempted to be moved outside the limit boundaries, the object will be constrained to those boundaries visually and when rendered but internally, its coordinates will still be changed beyond the limits. If the constraint is removed, the object will seem to jump to its internally specified location. Finally, if an object has an internal location that is beyond the limit, dragging the object back into the limit area will appear to do nothing until the internal coordinates are back within the limit threshold.

The limits for an object are calculated from its center.

Setting a min and max constraint to be the same value constrains the object's movement in that axis effectively locking changes to that axis while allowing motion in the other axis. Although this is possible, using the Transformation Properties axis locking feature is probably easier.

Options

minX, minY, minZ
: These settings specify the minimum location of the object's center in the x, y and z coordinate spaces. To place a limit, the minimum value in world coordinates of the object center location for each axis can be entered. The constraint will not happen unless the associated button for the axis is also pressed.

maxX, maxY, maxZ
: These settings specify the maximum location of the object's center in the x, y and z coordinate spaces. To place a limit, the maximum value in world coordinates of the object center location for each axis

can be entered. The constraint will not happen unless the associated button for the axis is also pressed.

Example

Previous: Manual/Constraints/Copy Scale Contents *Next:* Manual/Constraints/Limit Rotation

User Manual: Contents | Guidelines | Blender Version **2.44**

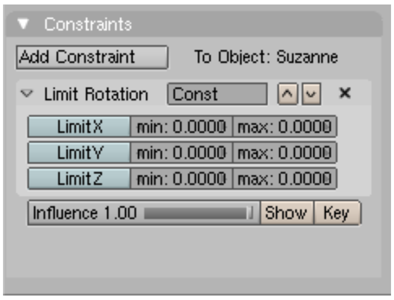
Limit Rotation Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



An Object can be rotated around the x, y and z axis. This constraint places a limit on the amount of rotation. A limit can be specified on the upper and lower values of the rotation around each of the axis. It is interesting to note that even though the limit constrains the visual and rendered rotation of the object, the object's data block still allows the object to have rotation values outside the minimum and maximum ranges. This can be seen in the Transform Properties N. When an object is rotated and attempted to be rotated outside the limit boundaries, the object will be constrained to those boundaries visually and when rendered but internally, its rotation values will still be changed beyond the limits. If the constraint is removed, the object will seem to jump to its internally specified rotation. Finally, if an object has an internal rotation that is beyond the limit, rotating the object back into the limit area will appear to do nothing until the internal rotation values are back within the limit threshold.

Setting a min and max constraint to be the same value constrains the object's rotation in that axis effectively locking changes to that axis while allowing rotation in the other axis. Although this is possible, using the Transformation Properties axis locking feature is probably easier.

Options

LimitX
: The minimum and maximum values of rotation around the x axis. The constraint will not happen unless the associated button for the axis is also pressed.

LimitY
: The minimum and maximum values of rotation around the y axis. The constraint will not happen unless the associated button for the axis is also pressed.

LimitZ

The minimum and maximum values of rotation around the z axis. The constraint will not happen unless the associated button for the axis is also pressed.

Example

Previous: Manual/Constraints/Limit Location	Contents	*Next:* Manual/Constraints/Limit Scale

User Manual: Contents | Guidelines | Blender Version **2.44**

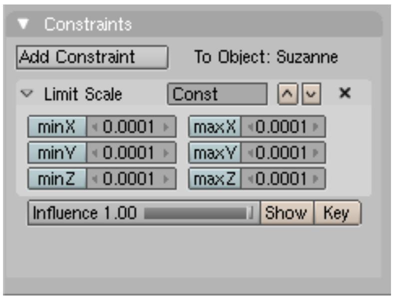
Limit Scale Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Options

minX, minY, minZ
foo
maxX, maxY, maxZ
bar

Example

Previous: Manual/Constraints/Limit Rotation	Contents	*Next:* Manual/Constraints/Track To

User Manual: Contents | Guidelines | Blender Version **2.44**

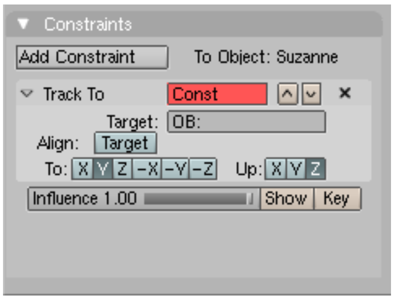
Track To Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Track To rotates an object to point at a target object. This constraint also forces the object to be "right−side" up. You can choose the positive direction of any of the three axes to be the up−side. This constraint shares a close relationship to the IK constraint in some ways. This constraint is very important in rig design, and you should be sure to read and understand the page on tracking, as it centers around the use of this, and the IK, constraints.

Options

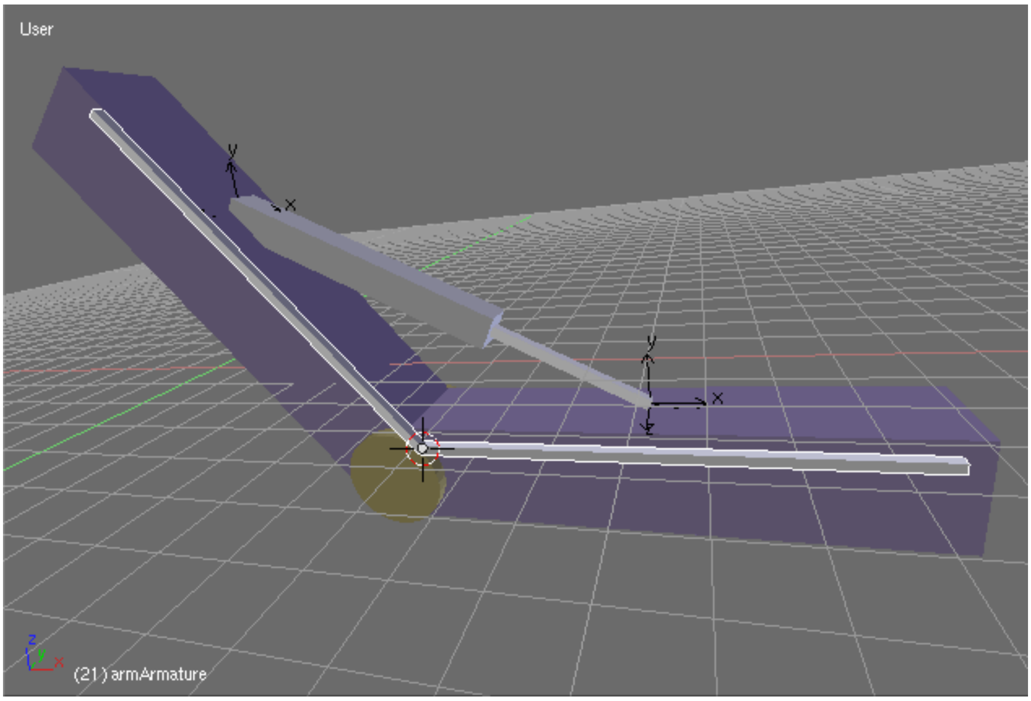
To
: The axis of the object that has to point to the target.

Up
: The axis of the object that has to be aligned (as much as possible) with the world Z axis. An Align: Target button, when enabled, uses the coordinates of the target object's Z axis, thus tilting or rocking the object as it tracks the target.

Head/Tail − With Bone targets only
: A number from 0.0 to 1.0 that represents the place on the target bone to Track to (0.0 = the bone's root; 1.0 = the bone's head)

Example

Piston



A piston rig created with Track To constraint.
Sample blend file

Previous: Manual/Constraints/Limit Scale Contents *Next:* Manual/Constraints/Floor

User Manual: Contents | Guidelines | Blender Version **2.44**

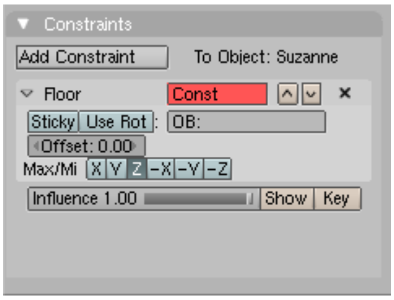
Floor Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Animation Tip:

When you animate foot placement on the floor plane, ***always*** be sure to use the option **VisualLoc** from the **Insert Key** menu, or enable **Use Visual Keying** from the **Auto Keyframing** menu in the User Preferences.

The Floor Constraint allows you to use a target object to specify the location of a plane which the affected object cannot pass through. In other words, it creates a floor! (or a ceiling, or a wall). This only works by default with planes of the global coordinate system.

Options

Sticky
: Makes the affected object immovable when touching the plane (cannot slide around on the surface of the plane), which is fantastic for making walk and run animations.

UseRot
: Takes the target object's rotation into account.

Offset
: A number of BU's from the object's center. Use this to account for the distance from a foot bone to the surface of the foot's mesh.

Max/Min
: Which axis is the floor? Things normally stick "down" Z, but can walk on walls as well.

Head/Tail − With Bone targets only
: A number from 0.0 to 1.0 that represents the place on the target bone to use for the Floor (0.0 = the bone's root; 1.0 = the bone's head)

Example

Previous: Manual/Constraints/Track To | Contents | *Next:* Manual/Constraints/Locked Track

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.44**

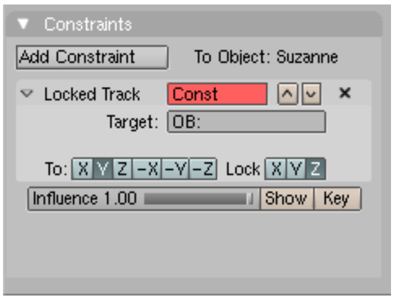
Locked Track Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Locked Track is a difficult constraint to explain, both graphically and verbally. The best real−world example would have to be a compass. A compass can rotate to point in the general direction of its target, but it can't point *directly at* the target, because it spins like a wheel on an axle. If a compass is sitting on a table and there is a magnet directly above it, the compass can't point to it. If we move the magnet more to one side of the compass, it still can't point *at* the target, but it can point in the general direction of the target, and still obey its restrictions of the axle.

When using a Locked Track constraint, you can think of the target object as a magnet, and the affected object as a compass. The "Lock" axis will function as the axle about which the object spins, and the "To" axis will function as the compass needle. Which axis does what is up to you! If you have trouble understanding the buttons of this constraint, read the tool−tips; they're pretty good. If you don't know where your object's axes are, turn on the **Axis** button in the **Draw** panel, object buttons, F7. Or, if you're working with bones, turn on the **Draw Axes** button, **Armature** panel, editing buttons, F9.

This constraint was designed to work cooperatively with the Track To constraint. If you set the axes buttons right for these two constraints, Track To can be used to point the axle at a target object, and Locked Track can spin the object around that axle to a secondary target.

This is all related to the topic discussed at length in the [Tracking](#) section.

Options

To
: The axis of the object that has to point to the target.

Lock

The axis of the object that has to be locked

Example

Previous: Manual/Constraints/Floor	Contents	*Next:* Manual/Constraints/Follow Path

User Manual: Contents | Guidelines | Blender Version **2.44**

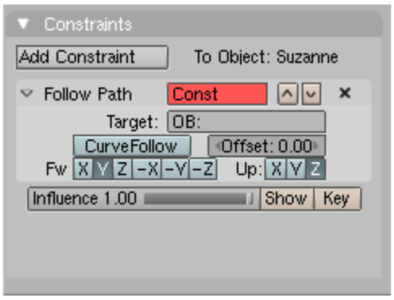
Follow Path Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Follow Path places the affected object onto a curve object. Curves have an animated property that causes objects along the path to move. In order for this to work, you have to have the CurvePath option activated (it's a property of the curve object). You can find it in the Curve and Surface panel in the Editing buttons, F9.

The movement along the path might be controled by two different ways: the most simple, in this same Curve and Surface panel, is to define the number of frames of the movement *via* the num button Path Len:, and its start frame *via* the constraint's Offset: option (by default: start frame **1** (= offset of **0**), duration **100**).

The second way – much more precise and powerful – is to define a Speed IPO curve for the path (Path section of the IPO Curve window). The start position along the path will correspond to an IPO value of **0.0**, and the end position, to an IPO value of **1.0**. You can therefore control the start frame, the speed of the movement, and the end frame, and even force your object to go forth and back along the path!

If you don't want objects on the path to move, you can give the path a flat speed IPO curve (its value will control the position of the object along the path).

Follow Path is another constraint that works well with Locked Track. One example is a flying camera on a path. To control the camera's roll angle, you can use a Locked Track and a target object to specify the up direction, as the camera flies along the path.

This constraint does not work well with bones.

Options

Offset
: The number of frames to offset from the "animation" defined by the path (by default: from the frame **1**).

CurveFollow

If this option isn't activated, the affected object's rotation isn't modified by the curve; otherwise, it's affected depending on the following options:

Fw

The axis of the object that has to be aligned with the forward direction of the path.

Up

The axis of the object that has to be aligned (as much as possible) with the world Z axis.
In fact, with this option activated, the behaviour of the affected object shares some properties with the one caused by a Locked Track constraint, with the path as "axle", and the world Z axis as "magnet"…

Example

…

Previous: Manual/Constraints/Locked Track	Contents	*Next:* Manual/Constraints/Clamp To

User Manual: Contents | Guidelines | Blender Version **2.44**

Clamp To Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description

Previously, we talked about Follow Path. Now lets talk about the Clamp To constraint.

The Clamp To constraint is a constraint which is particularly useful for moving things along large and complex paths which would otherwise be hard to hand−key smoothly. If you're feeling a little dÃ©jÃ vu, it's because the idea of a Clamp To constraint is very similar to Follow Path, with one main difference.

The difference is this: where Follow Path uses the time IPO of the curve that our constraint is targeting, Clamp To will get the actual position of the object(let's say a switch in some dark and Stygian spiral lock which must be moved back and forth in a combination, instead of the constant paths of the fighter and its pursuers in the example for the Follow Path constraint) and judge where to put the object(switch) by comparing the object's location to the curve it's targeting.

As with most things, of course, there's a bright side and a dark side.

The bright side is that when we're working with Clamp To, it will be easier to see what our object will be doing, since we're working in the 3D view port, in the same window that we're working with, it'll just be a lot more precise than sliding keys around on a time IPO and playing the animation over and over.

The dark(and Stygian) side is that, unlike in the Follow Path constraint, Clamp To doesn't have an option to track our object's rotation (pitch, roll, yaw) to the banking of the targeted curve, but−−like our lock, for example, or the armature example further below that−−we don't always need rotation on, so in cases like this it's usually a lot handier to fire up a Clamp To, and get the bits of rotation we do need some other way.

All together, what this means is that it'll probably be much easier to animate varied motion across a curve than it might be if we were using Follow Path, but even if it's not easier, it's an interesting alternative, so it can just come down to personal choice.

You don't need to know what Stygian means.

Options

Target

Text box − This is your basic target box, which just displays what your constraint is using as a reference. NOTE: In this case, the target object will have to be a curve object that the constrained object clamps to, so this box only works on curves.

Main Axis

This is a button group at the bottom − selecting one of these picks a global axis(x, y, z) for the constraint to use a reference to how far along the curve it's supposed to be. There's no real wrong choice, so just pick the axis that'd be easiest to work with or works best in your current situation−−a good idea is picking the axis which the targeted curve is the longest along, so that the global and constrained locations of the object will be more similar to each other−−or you can just pick Auto and Blender will make its best guess.

Example

The Dark and Stygian Lock.

Stygian Lock(object)

[.Blend file]

This is the Dark and Stygian lock that I've kept going on about, and now you all have to see it. Once again, you don't need to know what Stygian means, but before you all start throwing full wine bottles at your computer screens, looking at the picture on the right should give you a fair idea.

All the fancy design−work aside, though, the face and door are just decoration; the actual animation is done by the Clamp To constraint on the knob. If you watch, you'll see that the motion of it slides around the edge of the disk, goes back and forth, and even stops sometimes, and still manages to keep in a circle. This is the bright side of using a curve. Trying to key this the regular way would probably take much longer, and be much less smooth.

The bright side of, not just using a curve, but actually using a Clamp To constraint in this case−−instead of a Follow Path constraint−−happens around half−way through the animation. The knob slides all the way around, and then past the face's mouth, and keeps going a little while, before stopping and sliding back. Since the actual curve begins and ends in the figure's mouth, there needs to be an emergency key frame to jump the knob from being near the mouth at the end of the curve, to being near the mouth at the beginning. To do this, we have to visually move the location for both the keys as close to each other as possible, to make the transition smooth, this is something that would be much more difficult with the targeted curve's time IPO.

Note

I decided, in a spur of last−minute inspiration, that the camera would use a Clamp To constraint as well. It may seem a bit belabored, but we might as well go all the way and be consistent. It's a demo file, after all.

Arm(armature)

Previous: Manual/Constraints/Follow Path	Contents	*Next:* Manual/Constraints/Stretch To

User Manual: Contents | Guidelines | Blender Version **2.44**

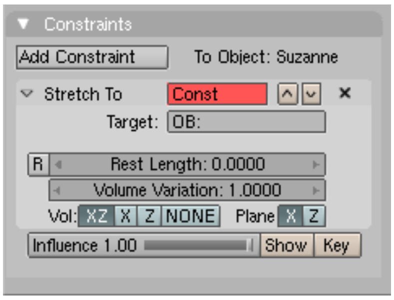
Stretch To Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



Stretch To causes the affected object to scale the Y axis towards a target object. It also has volumetric features, so the affected object can squash down as the target moves closer, or thin out as the target moves farther away. Or you can choose not to make use of this volumetric squash−'n'−stretch feature, by pressing the **NONE** button. This constraint assumes that the Y axis will be the axis that does the stretching, and doesn't give you the option of using a different one because it would require too many buttons to do so.

This constraint affects object orientation in the same way that Track To does, except this constraint bases the orientation of its poles on the original orientation of the bone! See the page on Tracking for more info. Locked Track also works with this constraint.

Options

R
: Pressing the R button calculates the rest length as the distance from the centers of the constrained object and its target

Rest Length
: Rest Length determines the size of the object in the rest position

Volume Variation
: Volume Variation controls the magnitude of the effect

Vol
: The Vol: buttons control along what axes the volume is preserved (if at all)

Plane
: The Plane buttons define which local orientation should be maintained while tracking the target

Example

Previous: Manual/Constraints/Clamp To Contents *Next:* Manual/Constraints/Rigid Body Joint

User Manual: Contents | Guidelines | Blender Version **2.44**

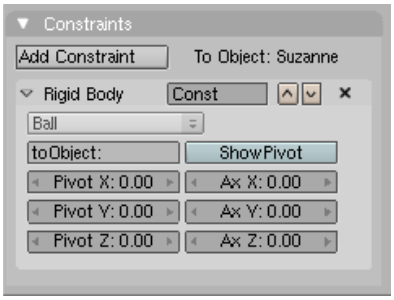
Rigid Body Joint Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



This constraint is mainly useful when using the game engine.

Options

Joint Types
: foo

toObject
: foo

ShowPivot
: foo

Pivot X, Y, Z
: foo

Ax X, Y, Z
: foo

Example

Previous: Manual/Constraints/Stretch To	Contents	*Next:* Manual/Constraints/IK Solver

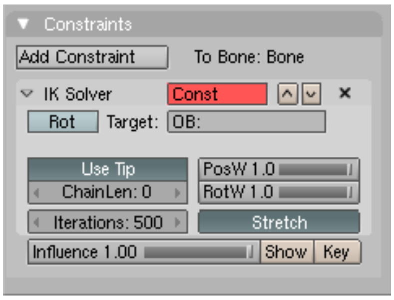
IK Solver Constraint

Mode: Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description

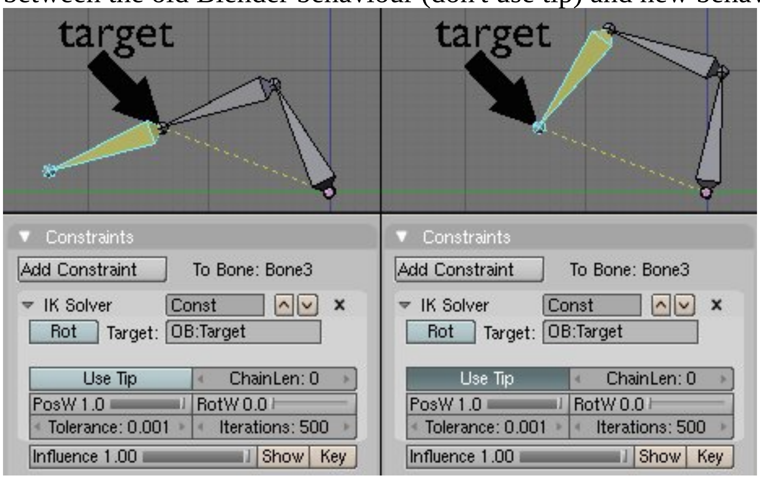


The IK Solver constraint is Blender's implementation of inverse kinematics. You add this constraint to a bone and then it, and the bones above it, become part of the inverse kinematic solving algorithm.

Options

Rot
: Toggle option to make the IK chain follow the rotation of the target object.

Use Tip
: Toggle option to use the tip of the bone instead of the base to solve the IK to. This option toggles between the old Blender behaviour (don't use tip) and new behavior (use tip).



An IK constraint on the yellow bone with indicated target. Left: without Use Tip. Right: with Use Tip.

ChainLen
: The number of bones above this bone that you want to be affected for IK. The default is 0, which means *all* bones above this bone will be used for IK.

PosW
: foo

RotW
: foo

Tolerance
: foo

Iterations
: foo

Example

Previous: Manual/Constraints/Rigid Body Joint	Contents	*Next:* Manual/Constraints/Action

 Action Constraint

User Manual: Contents | Guidelines | Blender Version **2.44**

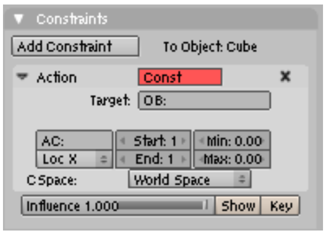
Action Constraint

Mode: Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



The Action constraint allows you to map any action to one of the rotation axes of a bone.

Options

Target
: the object to use in the constraint.

AC
: The action containing the keys for the specified bone.

Loc
: Choose which transformation axis (from the object) is used to set keys.

Start
: Starting frame of the action

End
: Final frame of the action.

Min
: The minimum value for the target channel range.

Max
: The maximum value for the channel range.

C Space
: The space (Global or Local) that the object is evaluated in.

Example

Previous: Manual/Constraints/IK Solver | Contents | *Next:* Manual/Constraints/Script

User Manual: Contents | Guidelines | Blender Version **2.46**

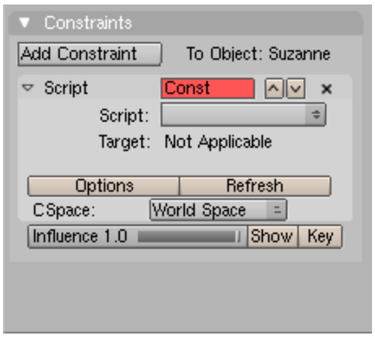
Script Constraint

Mode: Object Mode and Pose Mode

Panel: Object Context Constraints

Hotkey: F7

Description



The script constraint allows a person to write a new constraint, all in Python. A Pyconstraint can have multiple targets (however many the script requires), and even their own options.

Options

Script
: Sets the Pyconstraint to use.

Target
: Allows you to select what objects the script will target.

Options
: Change some of the constraint's settings.

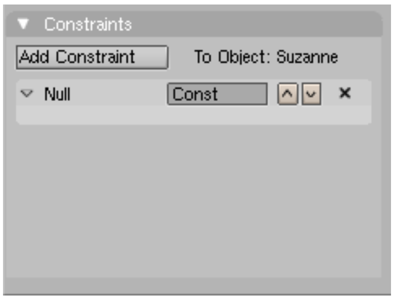
Refresh
: Forces the scripted constraint to refresh it's settings.

C Space
: The space (Global, Local) which the target is used.

Example

Previous: Manual/Constraints/Action	Contents	*Next:* Manual/Constraints/Null

User Manual: Contents | Guidelines | Blender Version **2.44**



The null constraint doesn't do anything. It is an antiquated feature.

Previous: Manual/Constraints/Script	Contents	*Next:* Manual/Character Animation

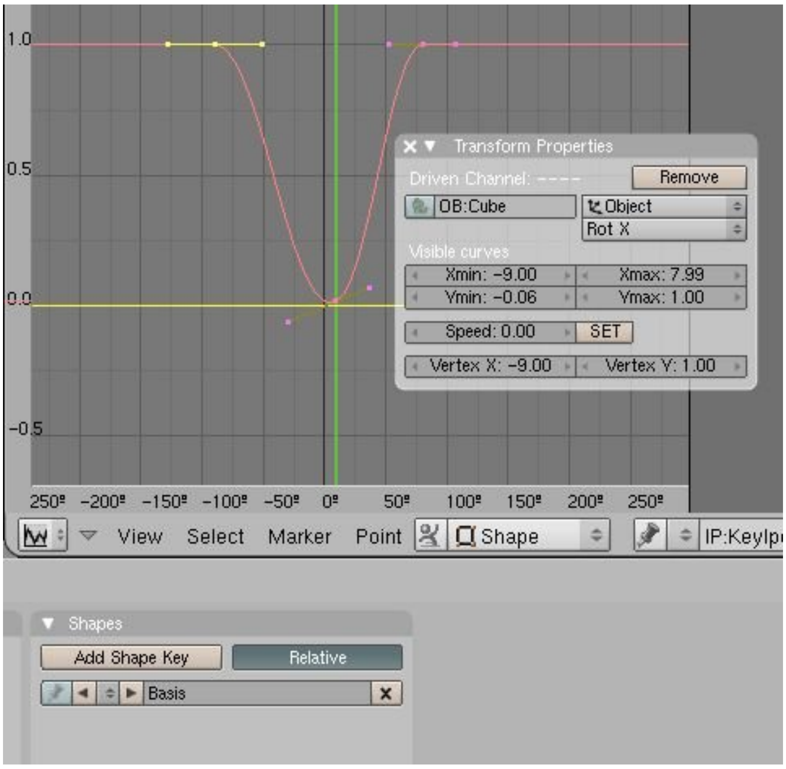
User Manual: Contents | Guidelines | Blender Version **2.45**

Advanced Animation

Driving Shape Keys

A shape key can be made to happen by something happening to another object. Animating that other object causes the shape key to exert its influence on the mesh. This other object can be any kind of object, but is commonly a mesh object that has been modeled to be be a custom shape that indicates what it does. For example, a mesh that controls the eyelids of a face may be a crescent shape.

As an analogy, think of a robot on a platform, with a control panel nearby. As you move a lever on the control panel, the servos in the robot activate and the robot moves its arm. In this example, the driver is the lever, and the shape key is the robot's arm. By moving or rotating or scaling the driver, the shape key activates.



The relationship between the change in the driver's location/rotation/scale and the influence on the shape key is through the Shape's IPO curve. With the base object's shape key selected,

In this example, if Cube is rotated about the X axis by less than −90 degrees or more than 90 degrees, the Basis shape will have 100% influence. For values between −90 and 90, the influence changes from 1 to 0 and back up to 1 again. If the RotX of the cube is 0, the basis shape has no influence. As the cube rotates from 0 to 90, the Basis shape will exert more and more influence.

Now, in your animation, add IPO keys for the Cube, and rotate it about the X axis. As you do, the shape key on the other mesh will activate and deform the mesh. Keep the driver mesh out of camera view, or on a hidden layer, or transparent so that it does not render.

You can see that a single driver object can drive up to 9 different shape keys individually; one for each LocXYZ, RotXYZ and ScaleXYZ

Previous: Manual/Non Linear Animation	Contents	*Next:* Manual/PartIX/Driven Shape Keys

User Manual: Contents | Guidelines | Blender Version **2.41**

A *Driven Shape Key* is the controlling of the *Key Value* of a relative *Shape Key* by the movement of another object.

If one would like to animate relative *Shape Keys* in Blender, this unfortunately, cannot be done with **Actions**. (See Action Editor)

Instead one uses an "auxiliary construction", Ipo Drivers. With an Ipo Driver you control the value of an Ipo curve by moving another object. So e.g. the color of an object can be dependent on the rotation of another object. Equally the *Key Value* − the influence strength − of a shape can be dependent on the rotation or movement of another object. This is especially useful, because one can make e.g. the form of a muscle dependent on the amount of rotation of the underlying bone.

Driven Shape Keys are a typical Blender function: built up from several elements, which unfold their power only in the correct combination. In this case that is the vertex animation by relative Shape Key, and the composition of actions in the NLA for Armatures. We do not have to use Armatures (and/or Bones) as "drivers" for the Shape Keys necessarily, but the animation possibilities with Poses are best. Therefore the entire functionality is already described in other sections, it's just that it's not too obvious to combine the elements.

By using driven Shape Keys you're working on increasingly higher levels of abstraction.

- The simplest thing is to work on the level of the Shape Keys, like "Left Eyebrow Down", "Right Eye Open", "Sneer Left".
- The next level is to drive one or more Shapes with the same armature bone, like "Eyelid" + "Eyebrow". Or "Bulge muscle if arm is rotated".
- The next level is to combine the movement of several bones together in an action, like "Happy", "Sad", "Wink", "Frown" ...
- The last level would be to combine the actions in the *NLA Editor*.

You don't have to use these things, but they may make your life and workflow easier.

Example

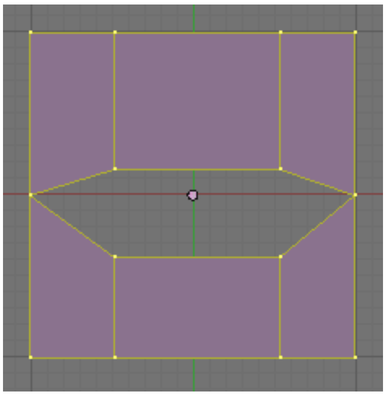


Image 1: Base Mesh for the Shape Keys example.

We will start with a simple example, which shows all essential elements.

The Basis Mesh is shown in *Image 1*. It is seen from Top−View (**Num−7**). Now we're going to insert five Shapes.

Select the mesh in Object mode.

First create the "Basis" shape by pressing **i**−>*Mesh*. (Or by by pressing the Add Shape Key button under the Shapes tab in the Edit/Buttons window).

- ♦ Make sure that you have created/edited the mesh to your liking **before** you create the "Basis" shape. If you edit the mesh after creating shape−keys unpredictable results may occur **

We create the following four shapes directly at the beginning, so they all originate from the "Basis" shape. Repeat the **i**−>*Mesh* process four times.

You can switch between the shapes by selecting the respective shape in the *Shapes* panel in the *Editing* buttons (**F9**). Now we're going to change the shapes.

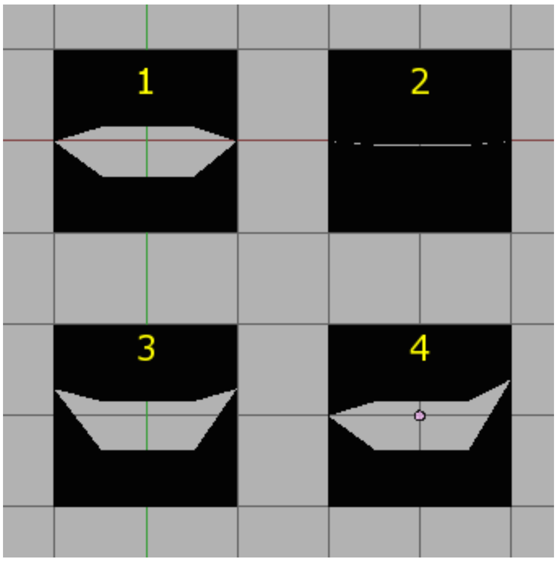


Image 2: Four of the five different shapes: Basis (1), Close (2), Smile (3), LeftUp (4).

The shapes:

1. Basis − Basis position, Mouth half opened.
2. Close − Mouth closed.
3. Smile − Corners of mouth pointing upwards.
4. LeftUp − Only left corner upwards.
5. RightUp − Only right corner upwards.

See the images −−−−−−−−−−−−>>>>>>>>>

To drive the Ipo curves we use the bones of an armature. One bone to open and close the mouth, another one to lift or lower the corners of the mouth. Since we can use all three directions in space separately as drivers input, we can move both corners of the mouth by moving in vertical direction, by moving in horizontal direction we move the right resp. left side only, etc.

It's up to you how many bones you use and how you want to control the shapes, it depends on the amount of control and ease of use you would like to achieve.

In our example the armature is inserted in ZX (Front) view (**Num−1**). You may insert it on it's own layer and use a separate 3D window to move the bones around. You should align the bones along the global axles, it is easier than to confine their movement.

Name the left bone "Close", the right bone "Smile". Name the armature object "Driver" (**F9**−>*Link and Materials* panel, *OB:* field.). (The name "Driver" is arbitrary and has no relation to the IPO−>"Driver")

Select the mesh in object mode.

Split the 3D window and change to the *Ipo Curve editor*. Select the *Ipo Type−>Shape* (*Image 3*).

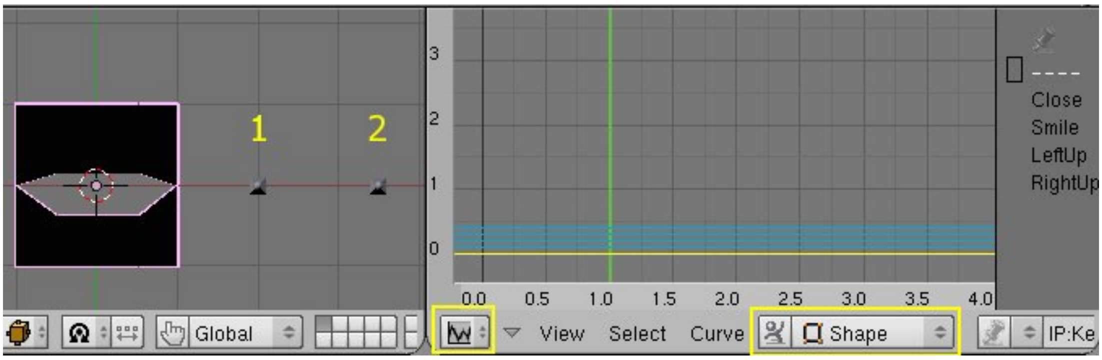


Image 3: Setup of the Armature. On the right side the Ipo window in *Shape* mode. We're looking at the bones bottom−up. 1: "CLOSE", 2: "Smile".

Connecting Shape key and Driver

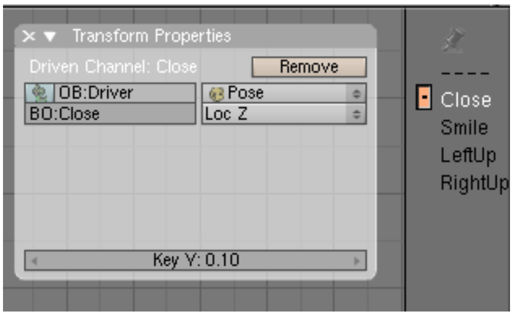


Image 4: A Bone as Ipo key Driver in the Ipo Curve editor.

To connect the Shape key with the controlling object − the Driver − you select the Shape key (LMB from the list on the right side of the IPO window), and press **n** in the Ipo window in order to call the *Transform Properties* panel. Click on *Add Driver*. Insert the name of the armature in the *OB:* field and select *Pose* in the drop down box. Insert the name of the controlling bone in the field *BO:* (here "Close"). We use *LocZ* to control the pose (*Image 4*). If you're in doubt about the correct coordinate, select the bone in the 3D window and use it's transform properties panel in the 3D window to watch it's coordinates.

To insert an Ipo curve press **i** in the Ipo Curve editor and confirm *Default one−to−one mapping*. A linear curve transition from (0,0) to (1,1) is inserted. The symbol for the Ipo curve in front of the shapes name receives a point, in order to indicate that a Driver for the curve is present. The Ipo curve maps the movement of the bone to the *Key value* of the shape, e.g. if you move the bone one Blender Unit (BU) in Z direction, the *Key value* of the shape changes from 0 to 1. If you move in negative Z direction the shape is inverted, i.e. the vertices move in opposite direction. If the Ipo curve runs horizontally the shape does not change if you move the bone.

We will use that to specify the borders of a meaningful movement of the bone.

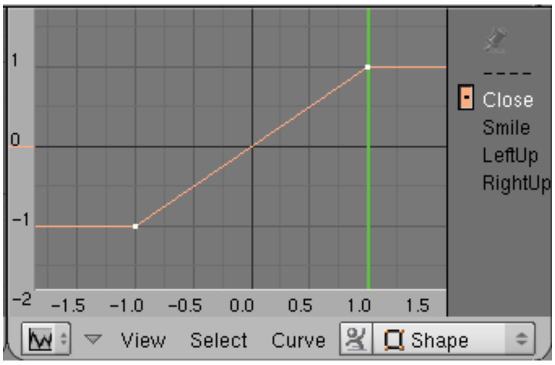


Image 5: Ipo curve for the shape "Close".

Set the mapping to (−1, −1), (1, 1) by selecting the curve and pressing Tab to enter edit mode. Right click on the point at (0,0), press G to grab it, and move your mouse down and to the left until the point has shifted to (−1,−1). You also can, once the point is selected, directly type its new coordinates in the fields Vertex X: and Vertex Y:, at the bottom of the Transform Properties pannel. Press Tab again to leave edit mode. Set the Extend mode to *Constant* (*Curve−>Extend mode−>Constant*). Press **T**−>*Linear* reset to a linear curve, if the type of the curve changes when the extend mode changes.

Repeat these steps for the shape "Smile", but we have to invert the Ipo curve (**S−X**) from (−1,1) to (1,−1), so that the movement of the bone corresponds with the movement of the mouth. In order to raise the corners of the mouth individually, the Shape key is connected with the x−coordinate of the Bone. If the Bone moves in positive x−direction the right corner is to be raised, if it moves in negative y−direction the left corner (*Image 6*).

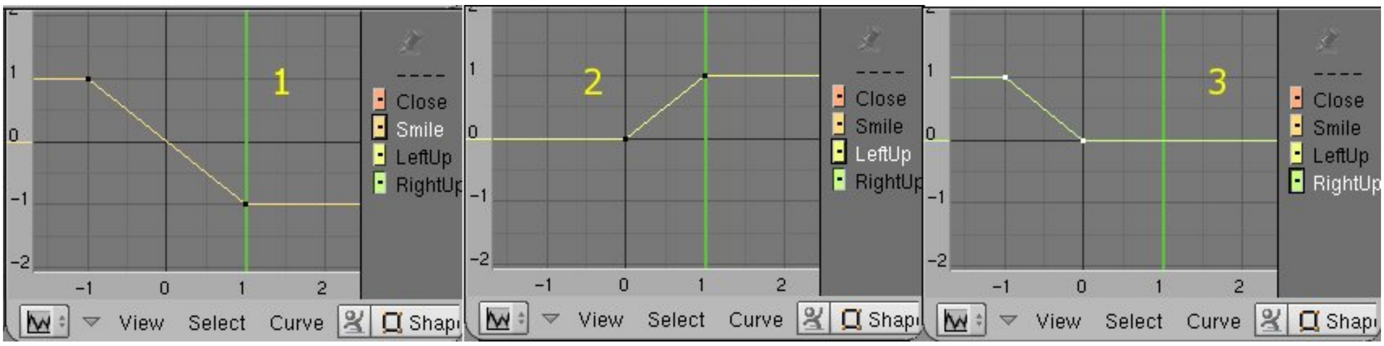


Image 6: The three Ipo curves for the Shapes "Smile" (1), "LeftUp" (2) and "RightUp" (3).

If you now move the bones in pose mode, the shapes change accordingly. If you need more shapes, or want the handling more flexible, you have to define more drivers.

Custom Controls

"Custom controls" − thus essentially selfmade manual controllers − facilitate the handling of Driven Shape Keys. There is no such Blenderobject (as of now), so we use mesh objects to visualise the purpose and borders of driver objects. The movement of the drivers is constricted by setting the lock options in their *Transform Properties* panel and − if you like − by floor constraints. So the driver objects behave pretty much like real control buttons.

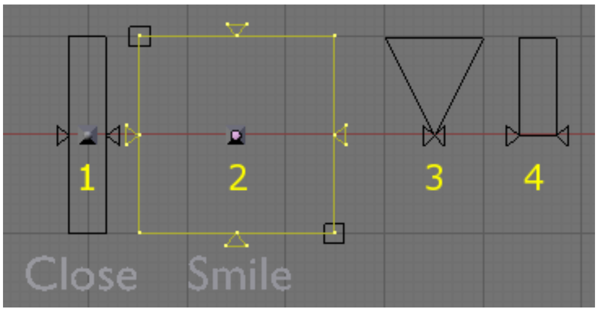


Image 7a: Four meshes to show the borders of the settings.

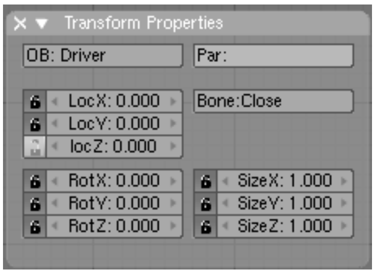


Image 8: *Transform Properties* panel for the bone "Close" (No. 1 in *Image 7a*.) The bone shall only move up and down.

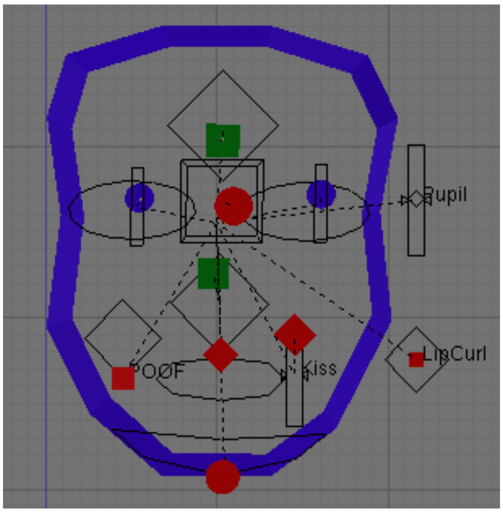


Image 7b: A more intuitive way to visualise the driver movements.

Image 7a shows some examples of Custom Controls (based on an idea by "PolygoneUK" from the Elysiun forum). By restricting the movement directions in the *Transform Properties* panel we allow the movement of the bone up and down, but not sideways.

You can go for a different visualisation if you build a stylized "Face", and insert the drivers in their respective places (idea by: Zeyne 2.4 rig with facial controller).

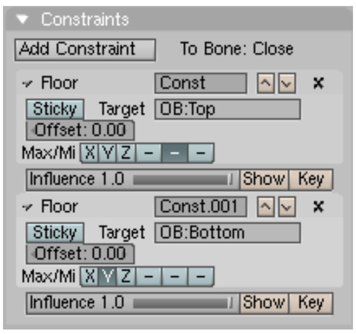


Image 9: If you use *Floor* Constraints you can confine the movements of the drivers to the area of the control.

In example 2 two further objects at the corners are used as *Floor* constraints. They prevent the movement of the bones beyond the indicated borders. However, one needs four constraints for an area, only two constraints for the "Up and Down" sliders (*Image 9*).

After creating the controls we can move them to a global scene, so you don't edit them inadvertently. Add a new, empty scene (the double arrow besides the *SCE:* field, *Add new−>Empty*). Name this new scene "Global". Mark all objects which belong to the control, but not the Armature, which we still have to change. Link the objects into the scene "Global" with Ctrl L−>*To scene... −>Global*.

Then we remove the connection between the objects: **U**−>*Object*. Now delete the objects in the current scene. Change into the *Scene* Buttons (**F10**) and click on the small Button with the double arrow and the inconspicuous Tooltip *Scene to link as a Set* in the *Output* panel. Select "global". Now the controls are indicated, but you can't select or edit them.

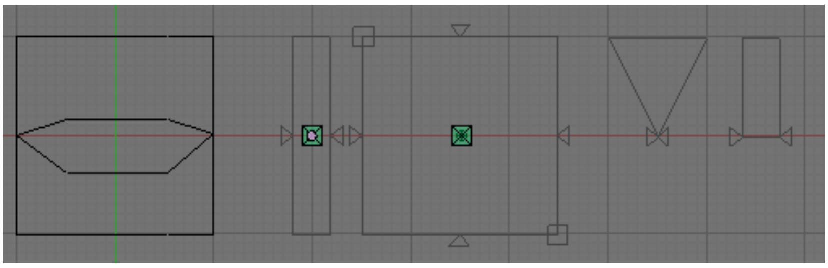


Image 10: The controls have been moved to a global scene.

NLA

Now the individual "face" movements can be arranged to actions, and these again in the NLA editor to complex groups. An action would be a certain face expression, e.g. merrily, sadly, surprised, bad etc. If one has provided all necessary actions, one works only with these actions.

Of course your're not forced to work in the NLA editor, instead you could create one long action, into which you insert all Ipos.

Define Actions

As described in the section [The Non Linear Animation Editor Window](), the different actions for the face have to be created.

Image 11: Three actions in the *Action Editor* window.

The example "face" is extended by two "eyes". Three actions were created: "Wink", "Joy" and "Anger" (*Image 11*). By positioning the bones of the armature in pose mode the expressions are created, all necessary bones selected and an Ipo inserted. Move ten frames foreward, and save exactly the same pose. The timing is made in the *NLA Editor* window.

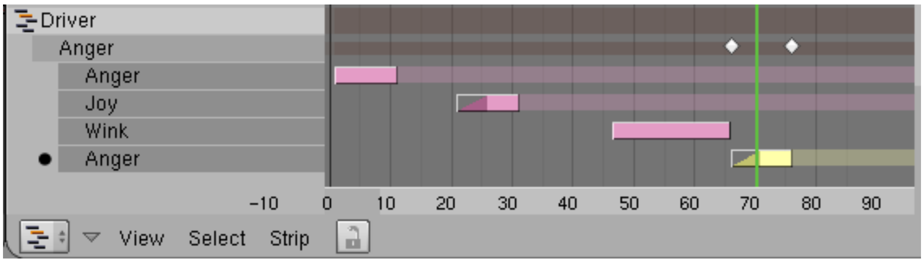


Image 12: The actions in the *NLA Editor* window.

Now arrange the expressions in the NLA editor and define the timing. Especially useful is the blending of different expressions (*Blendin/Blendout*). If you use the option *Hold* for the strips, you should define an action "Basis", so that you can blend to the "Basis" shape.

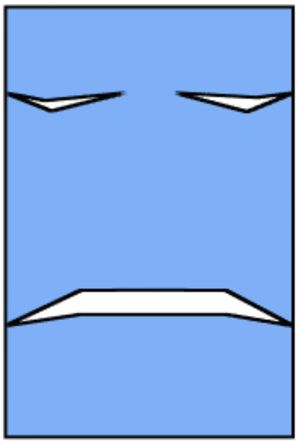


Image 13: Enjoy Blending!

Links

Ipo Drivers, Releasenotes of v2.40
Orange Blog: License to drive Driven Shape Keys, blend file provided.
Using Ipo driven shape keys to correct deformations in joints

Previous: Manual/PartIX	Contents	*Next:* Manual/PartIX/The Non Linear Animation Editor Window (NLA)

New (March 29, 2007)

Expressions can now use the name "self" to reference the object they belong to. Example: `self.LocY`. This should be available in Blender versions > 2.43.

Pydrivers: Python Ipo Drivers

Ipo Drivers

About
: "An IpoDriver is like an IpoCurve, but instead of a Bezier curve, it allows to connect a property of other Objects as input for the "channel". For example, IpoDrivers can be used to have a Shape Key being "driven" by the rotation of a Bone. Or the RGB colors of a Material get driven by the XYZ location of an Object."

Where they are
: "Editing of Drivers happens in the IpoWindow. Here you can notice that the channels (...) now have an "active" channel indicator. To add a Driver, you have to use the "Transform Properties" Panel (Nkey). Here you can add or remove a Driver to the active channel, and use the buttons to fill in what kind of relationship you want to establish."

(excerpts taken from http://www.blender.org/cms/Ipo_Drivers.680.0.html maintained by Ton Roosendaal)

Pydrivers

About
: Pydrivers allow to use one−line Python expressions as input for a channel, instead of using a property of another object, like normal ipo drivers do. An expression in programming is any combination of symbols that can be evaluated to a definite value.

Where they are
: When adding a driver, like explained above, click on the small "Python" icon at the "Transform Properties" panel and a text input box will appear. Write your py expression there.

These drivers open up many interesting possibilities: we can use mathematical functions and more general programming to drive object animations. The only restriction is that the pydriver expression *itself* must return a real number when evaluated and not, e.g., a string or something else.

A simple example would be using sines or cosines to make objects oscilate around a given point, for example. Or (ugh) using random values at each animation frame to change some material attribute, like diffuse rgb, alpha, etc.

Note for Python programmers

Pydriver evaluation is equivalent to what the builtin eval() function does, with the restriction that pydriver expressions must return real numbers (any type of number that can be automatically cast to a float).

Instructions

Valid Expressions

We've already told the basics: there are text input boxes where you can type expressions in Python. Here are some examples of valid expressions:

- any real value: `1.0`
- expressions with numbers and operators: `4.5 + 8.9 * 7.0 - (2 / 3.0)`
- expressions also with variables: `math.pi * 2 + 5.0`
- available data: `Blender.Get("curframe") # the current animation frame`
- a little math: `math.sin(Blender.Get("curframe")) # the sine of the current frame!?`

Builtin resources and aliases

Pydrivers use their own global dictionary that is cached and only gets recreated when any expression fails.

In this dictionary we pre−import a few modules so they can be used inside pydriver expressions:

Note: to save typing and keep expressions smaller, we've added aliases for each module: Blender can be referenced as "Blender" or simply as "b". Below each module is followed by its available aliases:

- all from builtin (the default builtin module)
- Blender: blender, b
- Blender.Noise: noise, n
- math: math, m

Example expression: `m.cos(m.pi * b.Get("curframe") / n.random())`

Aliases were also added for a few commonly needed data:

- `ob(name)` is equivalent to `Blender.Object.Get(name)`
- `me(name)` is equivalent to `Blender.Mesh.Get(name)`
- `ma(name)` is equivalent to `Blender.Material.Get(name)`

Example expression: `ob("Cube").LocX + ob("Lamp").RotZ`

NEW (March 29, 2007): the name "self" was added to the pydrivers dictionary, as a way to easily access the object that owns a given expression.

Example expression: `self.LocX * 10`

The pydrivers.py Blender text

Besides the above modules, if there's a Blender text called "pydrivers.py" loaded in the Text Editor, it's also imported:

- pydrivers: pydrivers, p

This allows users to create their own functions and add their own variables without the restriction of the one−line py expression. For example, if your pydrivers.py text looks like this:

```
myvar = 10.0

def myfunction(arg):
  # do something fancy here
  return float_val
```

You can access both myvar and myfunction inside any expression:

Example expression: `p.myvar * p.myfunction(2) - 1`

Note: if you make updates to the pydrivers.py text, go to the Ipo window and click in any pydriver's text input box (in the Transform Properties panel), then click out of it or press ENTER, to force reloading the pydrivers.py module and to update all pydrivers at once.

Patch and sample .blend files

Entry in the patch tracker: Pydrivers

The patch there is against current cvs.

There are a couple of **lousy** .blend files there, too, demonstrating a few possibilities. The fishes.blend one was inspired by an artist's demo pointed to me by UnNamed (GSR, thanks!). Please check the README text in each .blend to get specific info about them.

TODO

- ~~Commit?~~ UPDATE: pydrivers are in CVS as of Sun April 30.
- UPDATE: clicking in and out of any pydriver's text input box is enough to at once reload "pydrivers.py" if available, re−evaluate all pydrivers (check console to see if errors were found) and update Blender's dependency graph (the DAG). Do we need to add an entry (like "Update PyDrivers") to some menu, too?
- As a test I disabled the restriction that pydriver expressions can't reference their own object (this is an ipo driver restriction related to the dependancy graph). If we really have to follow that restriction also with pydrivers, the code to force it just needs to be uncommented in EXPP_interface.c.

Links

- Check the docs in this very wiki's Main Page for Blender and Blender Python API references.
- Python and its documentation
- this might be a good hunting ground for those looking for functions to try with pydrivers: http://functions.wolfram.com/ (newcomers are recommended to start with elementary ones, specially trigonometric).
- Finally (and again), the patch tracker entry, with patch and sample .blend files for pydrivers: right here

User Manual: Contents | Guidelines | Blender Version **2.37**

Introduction

Blender's SoftBody system allows vertices to move based on the laws of physics. This mean that they can be set to react to gravity and wind. Objects in Blender can be set to be a soft body. Only *Mesh* and *Lattice* objects are implemented in release 2.37. The SoftBody system is primarily meant to enhance animation systems, including character animation. Effects like flexible or rippling skin are now very easy to achieve. In the 2.37 demo files (4 MB) you can find two examples of soft bodies, softbody_basics.blend and wind_soft.blend http://download.blender.org/demo/test/test237a.zip.

TODO: add something about fluids to this intro

Previous: BlenderDev/PyDrivers	Contents	*Next:* Manual/Particles

User Manual: Contents | Guidelines | Blender Version **2.45**

Particles

The particle system of Blender is fast, flexible, and powerful. Every Mesh−object can serve as an emitter for particles. The particles themselves can be mass−less, like Halos, or can be objects, like a coin or piece of confetti. These objects can be any type of Blender object, for example Mesh−objects, Curves, Metaballs, and even Lamps.

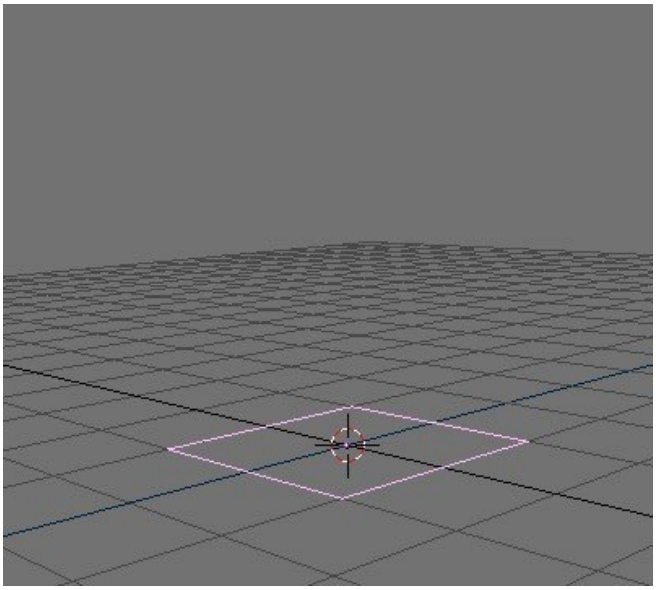
Particles can be influenced by a global force to simulate physical effects, like gravity or wind. With these possibilities you can generate smoke, fire, explosions, dust, fireworks or even flocks of birds. With hair particles you can generate fur, grass, and even plants.

Start from Frame 1:

To re−initialize the particle engine, it may be necessary to rewind all the way back to frame 1 and play forward after making changes to any settings or to the emitting mesh, so that Blender can re−calculate the particle motion.

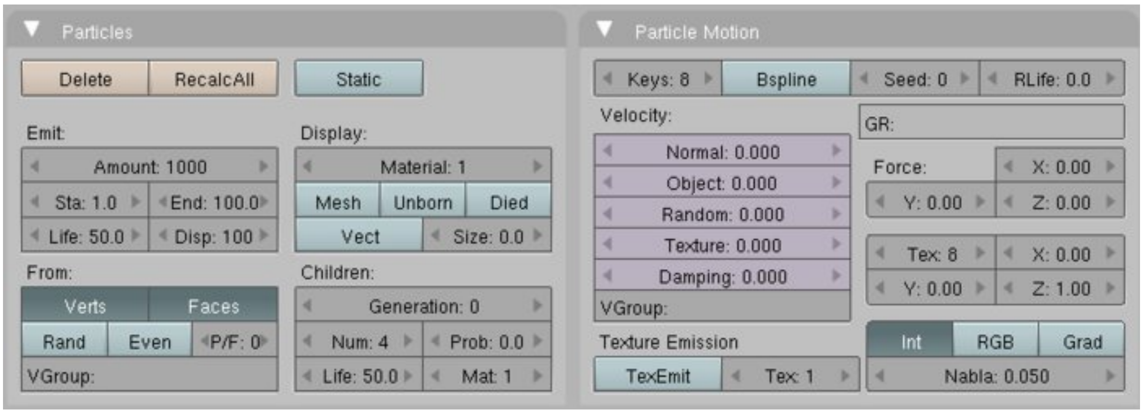
Mini−Tutorial: A first Particle System

Reset Blender to the default scene, or make a scene with a single plane added from the top view. This plane will be our particle emitter. Rotate the view so that you get a good view of the plane and the space above it (*The emitter*.).



The emitter.

Switch to the Physics Tab () in the Object Context (F7 or) and click the button NEW in the Particles tab, which then fills up with plenty of controls − and a new tab (*The Particle Buttons.*).



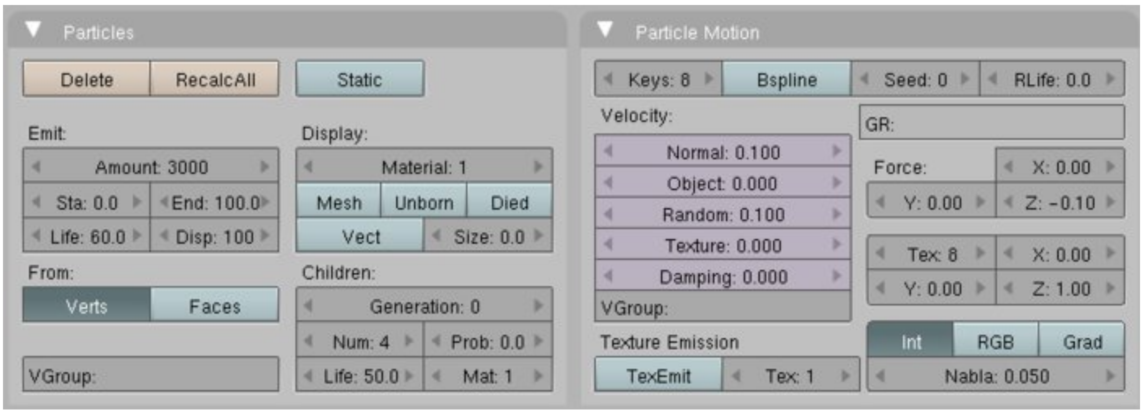
The Particle Buttons.

In the Particle motion tab, set the Normal: NumButton to 0.100 with a click on the right part of the button or use Shift LMB to enter the value from the keyboard. Play the animation by pressing Alt A with the mouse over the 3DWindow. You will see a stream of particles ascending vertically from the four vertices.

Congratulations − you have just generated your first particle−system in a few easy steps!

To make the system a little bit more interesting, it is necessary to get deeper insight on the system and its buttons (*Particles settings.*):

- The parameter Amount: controls the overall count of particles. On modern speedy CPUs you can increase the particle count without noticing a major slowdown.
- The total number of particles specified in the Amount: button are uniformly created along a time interval. Such a time interval is defined by the Sta: and End: NumButtons, which control the time interval (in frames) in which particles are generated.
- Particles have a lifetime, they last a given number of frames, from the one they are produced in onwards, then disappear. You can change the lifetime of the particles with the Life: NumButton.
- The Normal: NumButton used before (Particle motion tab) made the particles having a starting speed of constant value (0.1) directed along the vertex normals. To make things more "random" you can set the Random: NumButton to 0.1 too. This also makes the particles start with random variation to the speed.
- Still in the Particle motion tab, use the Force: group of NumButtons to simulate a constant force, like wind or gravity. A Force: Z: value of −0.1 will make the particles fall to the ground, for example.



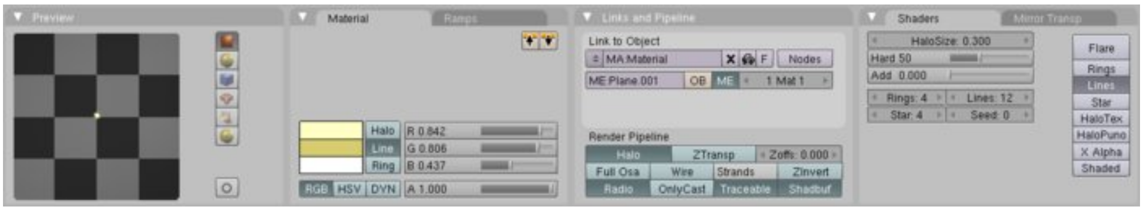
Particles settings.

This should be enough to get you started, but don't be afraid to touch some of the other parameters while you're experimenting. We will cover them in detail in the following sections.

Rendering a particle system

Maybe you've tried to render a picture from our example above. If the camera was aligned correctly, you will have seen a black picture with grayish blobby spots on it. This is the standard Halo−material that Blender assigns a newly generated particle system. Position the camera so that you get a good view of the particle system. If you want to add a simple environment, remember to add some lights. The Halos are rendered without light, unless otherwise stated, but other objects need lights to be visible.

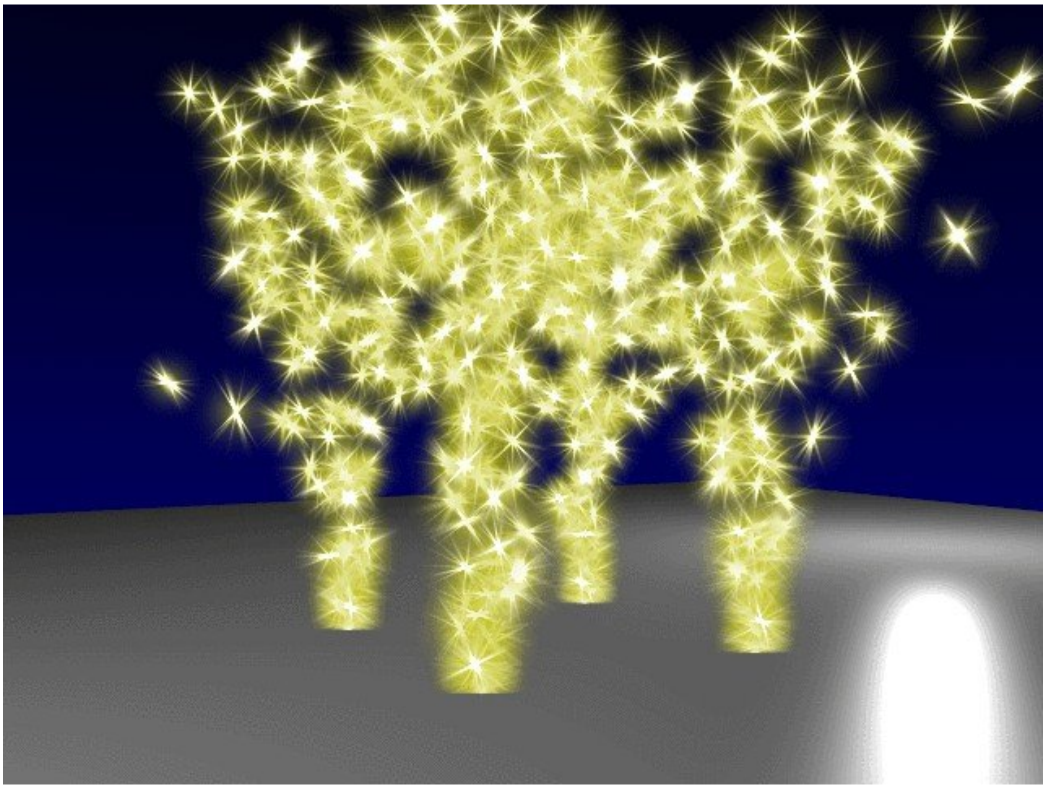
Go to the Material Buttons (F5) and add a new material for the emitter if none have been added so far. Click the Button "Halo" from the Links and Pipeline tab (*Halo settings*).



Halo settings

The Material Buttons change to the Halo Buttons. Choose Line, and adjust Lines: to a value of your choosing (you can see the effect directly in the Material−Preview). Decrease HaloSize: to 0.30, and choose a color for the Halo and for the Lines (*Halo settings*).

You can now render a picture with F12, or a complete animation (Ctrl F12) and see thousands of stars flying around (*Shooting stars*).

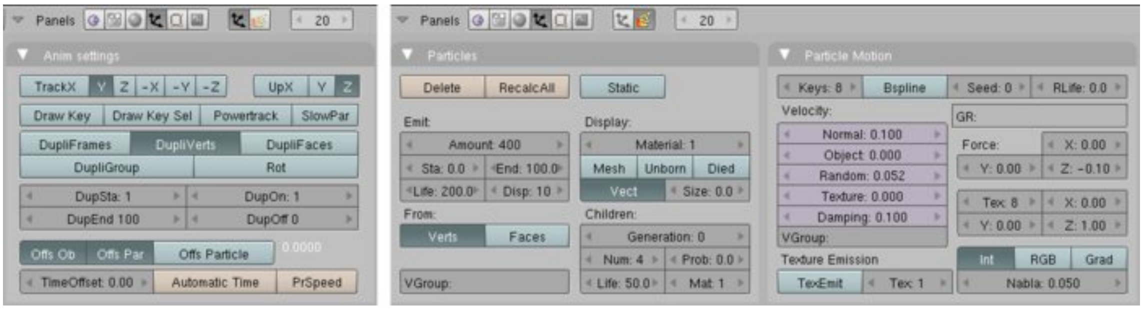


Shooting stars

Objects as particles

It is very easy to use real objects as particles, it is exactly like the technique described in DupliVerts. Start by creating a cube, or any other object you like, in your scene. It's worth thinking about how powerful your computer is, as we are going to have as many objects, as Amount: indicates, in the scene. This means having as many vertices as the number of vertices of the chosen object times the value of Amount:!

Scale the newly created object down so that it matches the general scene scale. Now select the object, then Shift RMB the emitter and make it the parent of the cube using Ctrl P. Select the emitter alone and check the option "DupliVerts" in the Anim Settings tab in the Object buttons, Objetc context (F7). The dupliverted cubes will appear immediately in the 3DWindow.

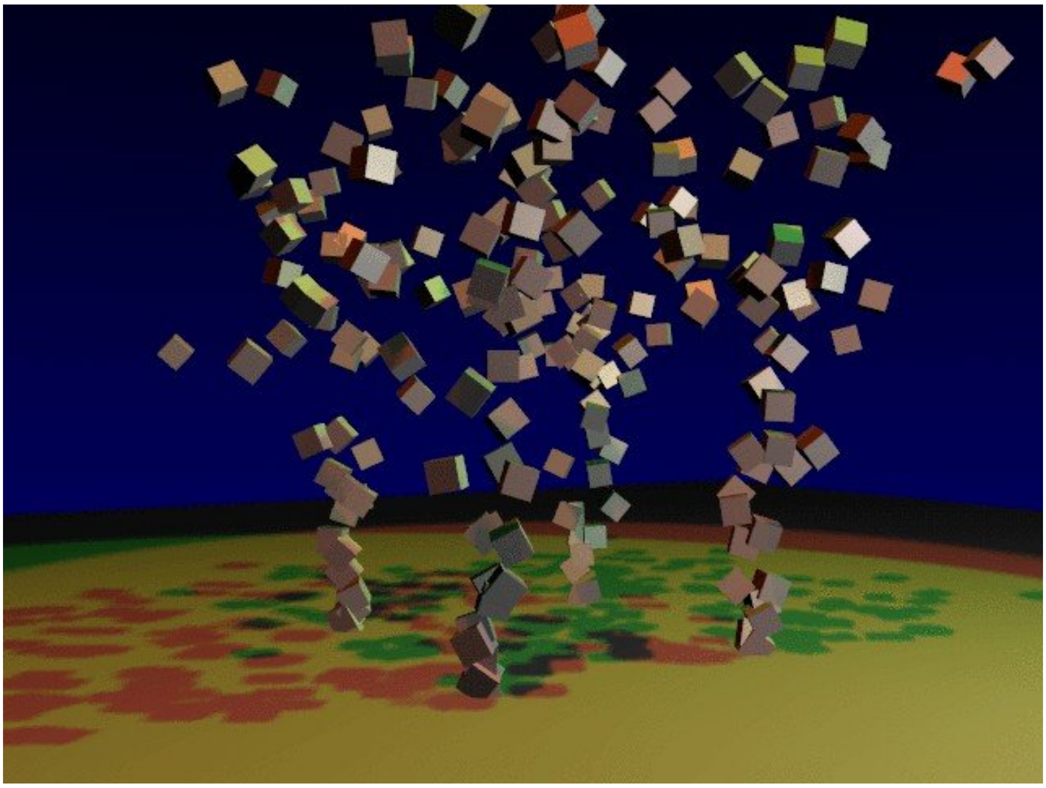


Setting Dupliverted Particles.

You might want to bring down the particle number shown in the 3D views before pressing Alt A: the numbutton Disp: allows you to set the percentage of particles to show in the 3D views (*Setting Dupliverted Particles.*). In the animation you will notice that all cubes share the same orientation. This can be interesting, but it can also be interesting to have the cubes randomly oriented. This can be done by checking the option Vect in the particle−parameters, which causes the dupli−objects to follow the rotation of the particles, resulting in a more natural motion (*Setting Dupliverted Particles.*). One frame of the animation is shown in (*Dupliverted particles rendering.*).

Original Object

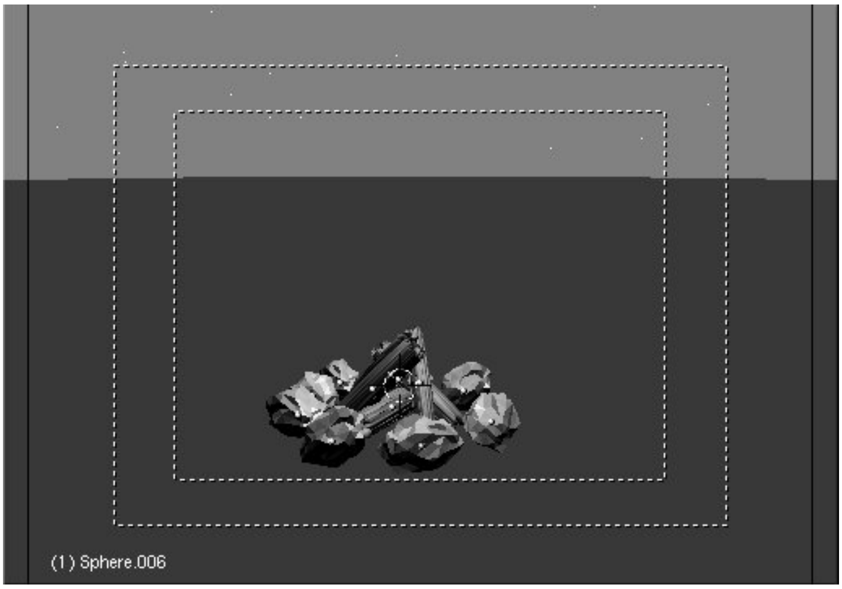
Take care to move the original object out of the camera view, because, differently than in regular Mesh Dupliverts, in Dupliverted particles it will also be rendered! −− This may have been true in older versions but in 2.45 this does not appear to be necessary; the dupliverted object does not show up in the render. If this can be confirmed then this Note can be removed.



Dupliverted particles rendering.

Making fire with particles

The Blender particle system is very useful for making realistic fire and smoke. This could be a candle, a campfire, or a burning house. It's useful to consider how the fire is driven by physics. The flames of a fire are hot gases. They will rise because of their lower density when compared to the surrounding cooler air (same principle as a hot−air balloon). Flames are hot and bright in the middle, and they fade and become darker towards their perimeter. Prepare a simple set−up for our fire, with some pieces of wood, and some rocks (*Campfire setup.*).

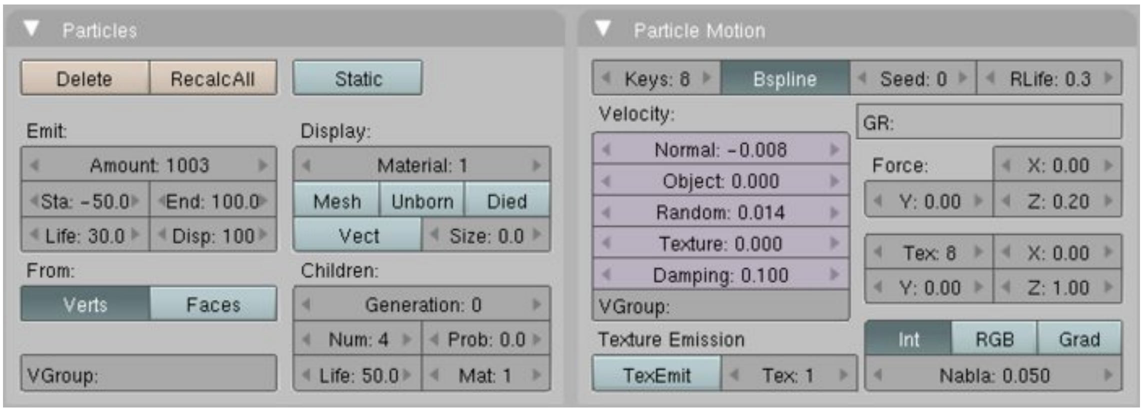


Campfire setup.

The particle system

Add a plane into the middle of the stone−circle. This plane will be our particle−emitter. Subdivide the plane once. You now can move the vertices to a position on the wood where the flames (particles) should originate.

Now go to the Object Context F7, Physics buttons, and add a new particle system to the plane. The numbers given here (*Fire particles setup*.) should make for a realistic fire, but some modification may be necessary, depending on the actual emitter's size.



Fire particles setup.

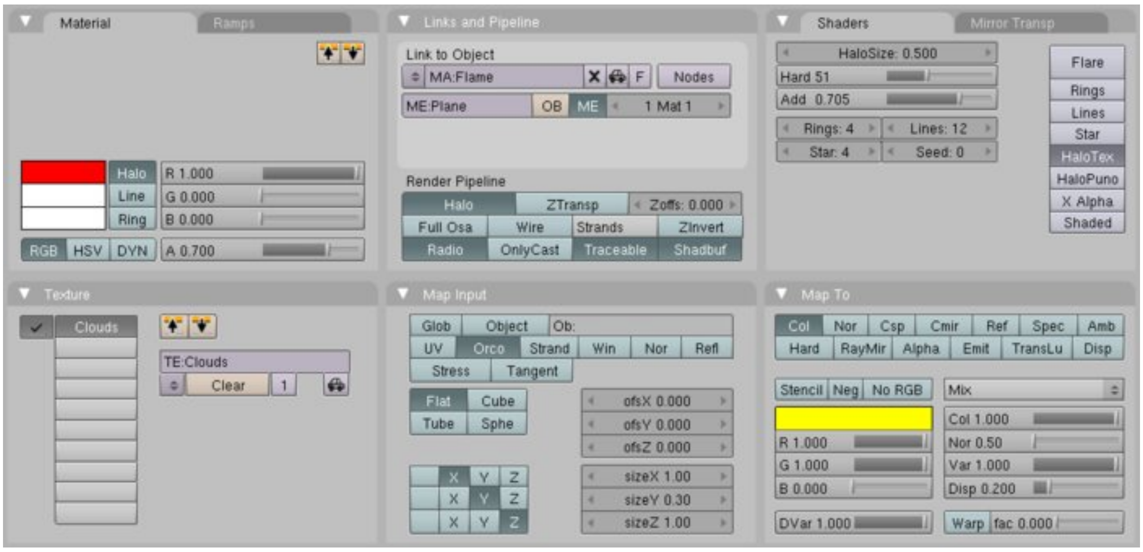
Some notes:

- To have the fire burning from the start of the animation make Sta: negative. For example, try −50. The value of End: should reflect the desired animation length.
- The Life: of the particles is 30. Actually it can stay at 50 for now. We will use this parameter later to adjust the height of the flames.
- Make the Norm: parameter a bit negative (−0.008) as this will result in a fire that has a bigger volume at its basis.
- Use a Force: Z: of about 0.200. If your fire looks too slow, this is the parameter to adjust.
- Change Damp: to 0.100 to slow down the flames after a while.
- Activate the Bspline Button. This will use an interpolation method which gives a much more fluid movement.
- To add some randomness to our particles, adjust the Random: parameter to about 0.014. Use the RLife: parameter to add randomness in the lifetime of the particles; a really high value here gives a lively flame.
- Use about 600−1000 particles in total for the animation (Amount:).

In the 3DWindow, you will now get a first impression of how realistically the flames move (Alt A). But the most important thing for our fire will be the material.

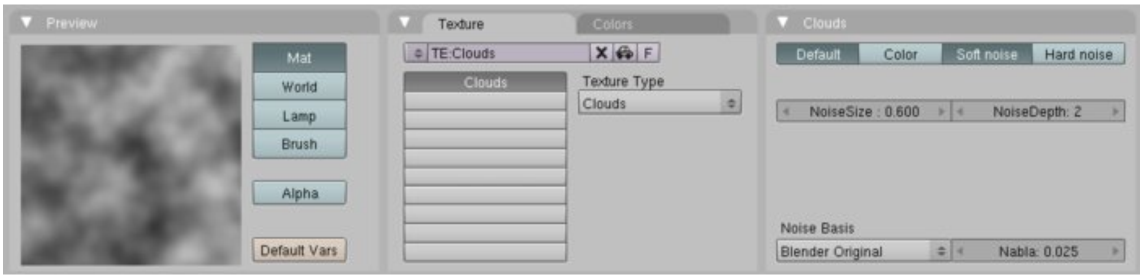
The fire−material

With the particle emitter selected, go to the Shading Context F5 and add a new Material. Make the new material a halo−material by activating the Halo button. Also, activate HaloTex, located in the shaders panel. This allows us to use a texture later.



Flames Material.

Give the material a fully saturated red colour with the RGB−sliders. Decrease the Alpha value to 0.700; this will make the flames a little bit transparent. Increase the Add slider up to 0.700, so the Halos will boost each other, giving us a bright interior to the flames, and a darker exterior. (*Flames Material.*).



Flames Texture.

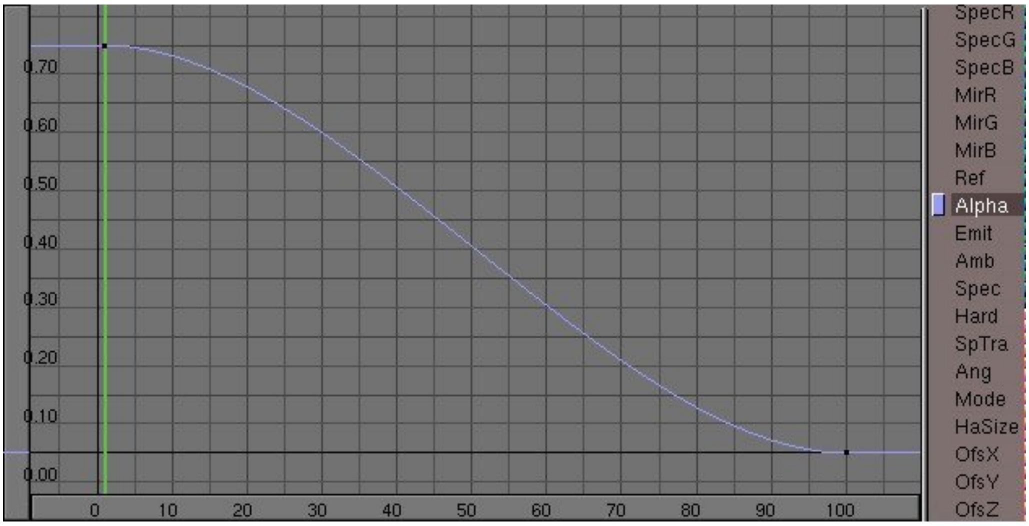
If you now do a test render, you will only see a bright red flame. To add a touch more realism, we need a texture. While the emitter is still selected, go to the Texture Panel and add a new Texture select the Clouds−type for it in the Texture (F6) Buttons. Adjust the NoiseSize: to 0.600. (*Flames Texture.*).

Go back to the Material Buttons F5 and make the texture colour a yellow colour with the RGB sliders in the Map To tab. To stretch the yellow spots from the clouds texture decrease the SizeY value down to 0.30. A test rendering will now display a nice fire. But we still need to make the particles fade out at the top of the fire. We can achieve this with a material animation of the Alpha and the Halo Size. Be sure that your animation is at frame 1 (Shift) and move the mouse over the Material Window. Now press I and choose Alpha from the appearing menu. Advance the frame−slider to frame 100, set the Alpha to 0.0 and insert another key for the Alpha with I. Switch one Window to an IPO Window. Activate the Material IPO Type by clicking the pertinent Menu Entry in the IPO Window header. You will see one curve for the Alpha−channel of the Material (*Fire Material IPO*).

Note

An animation for a particle material is always mapped from the first 100 frames of the animation to the lifetime of a particle. This means that when we fade out a material in frame 1 to 100, a particle with a lifetime

of 50 will fade out in that time. For objects mapped to particles through Dupliverts it seems that this animation technique does not apply/work. Any Alpha IPO added to the emitter seems to be ignored and an Alpha IPO seems to work against the absolute timeline and not anything relative to when the particle was emitted.



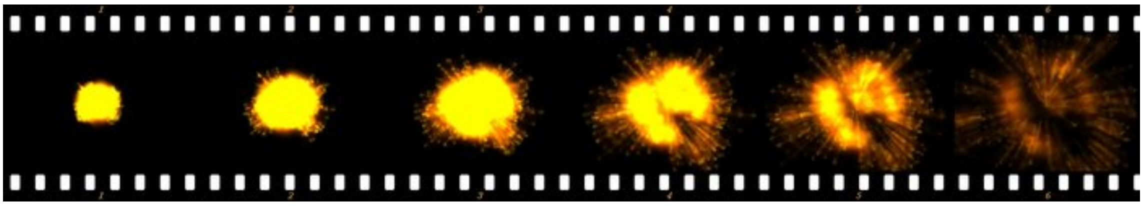
Fire Material IPO

Now you can render an animation. Maybe you will have to fine−tune some parameters like the life−time of the particles. You can add a great deal of realism to the scene by animating the lights (or use shadow−spotlights) and adding a sparks particle−system to the fire. Also recommended is to animate the emitter in order to get more lively flames, or use more than one emitter (*Final rendering.*).

Final rendering.

A simple explosion

This explosion is designed to be used as an animated texture, or for composing it with the actual scene. For a still rendering, or a slow motion of an explosion, we may need to do a little more work in order to make it look really good. But bear in mind, that our explosion will only be seen for half a second (*The explosion*).



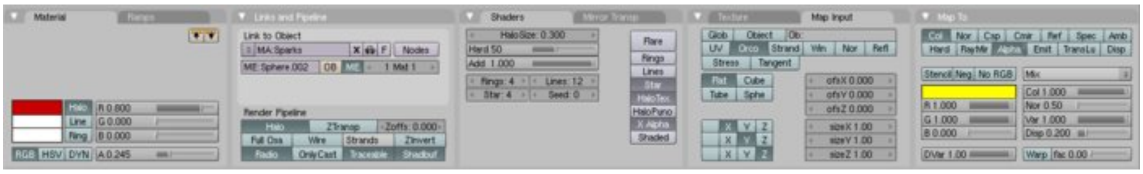
The explosion

As emitter for the explosion I have chosen an IcoSphere. To make the explosion slightly irregular, I deleted patterns of vertices with the circle select function in Edit Mode. For a specific scene it might be better to use an object as the emitter, which is shaped differently, for example like the actual object you want to blow up. My explosion is composed from two particle systems, one for the cloud of hot gases and one for the sparks. I took a rotated version of the emitter for generating the sparks. Additionally, I animated the rotation of the emitters while the particles were being generated.

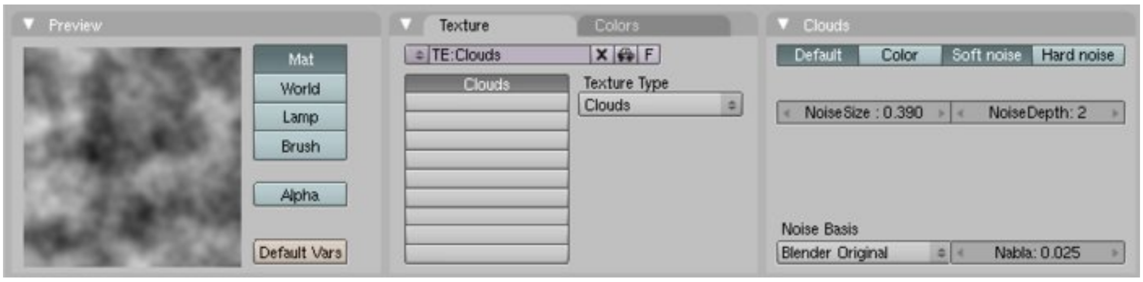
The materials

The particles for the explosion are very straightforward halo materials, with a cloud texture applied to add randomness, the sparks too have a very similar material, see *Material for the explosion cloud.* to *Texture for both.*.

Material for the explosion cloud.



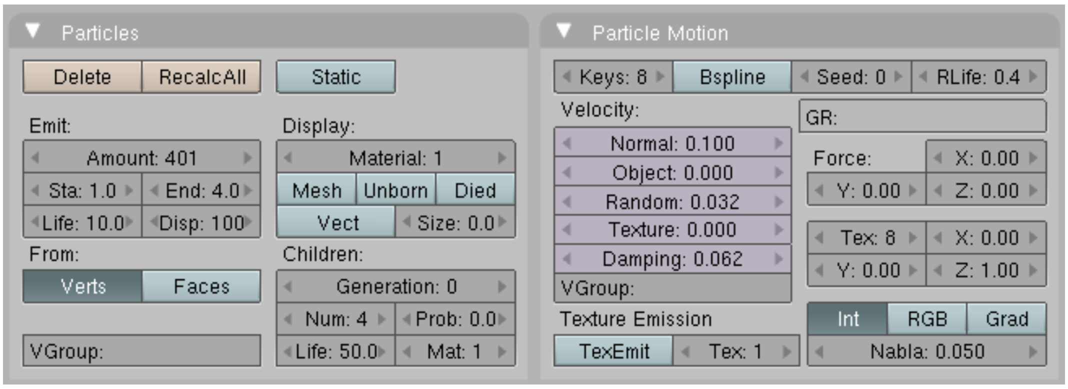
Material for the sparks.



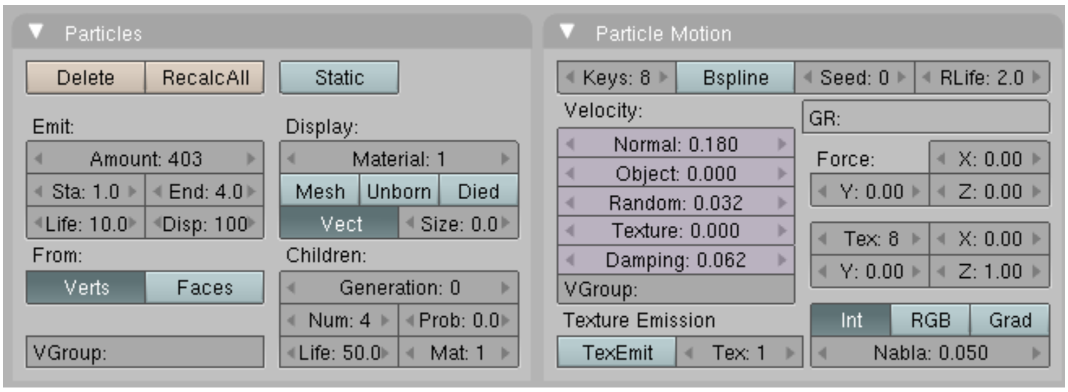
Texture for both.

Animate the Alpha−value of the Halo particles from 1.0 to 0.0 at the first 100 frames. This will be mapped to the life−time of the particles, as is usual. Notice the setting of Star in the sparks material (*Material for the sparks.*). This shapes the sparks a little bit. We could have also used a special texture to achieve this, however, in this case using the Star setting is the easiest option.

The particle−systems



Particle system for the cloud.



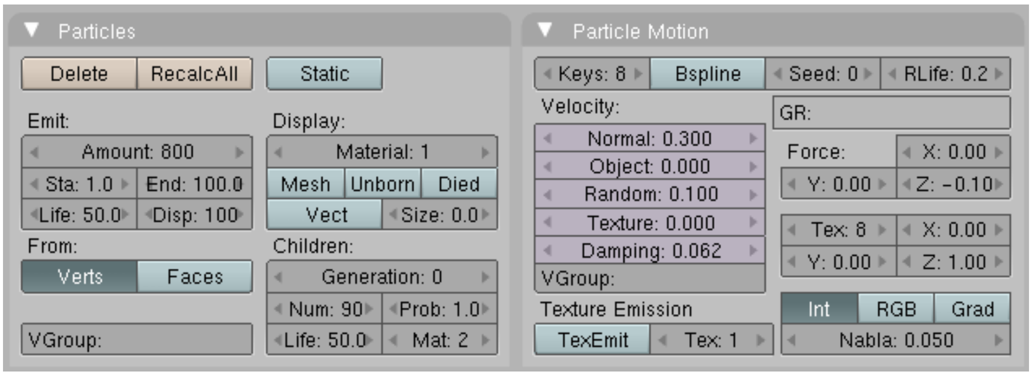
Particle system for the sparks.

As you can see in *Particle system for the cloud* and *Particle system for the sparks*, the parameters are basically the same. The difference is the Vect setting for the sparks, and the higher setting of Norm: which causes a higher speed for the sparks. I also set the RLife: for the sparks to 2.000 resulting in an irregular shape. I suggest that you start experimenting, using these parameters to begin with. The actual settings are dependent on what you want to achieve. Try adding more emitters for debris, smoke, etc.

Fireworks

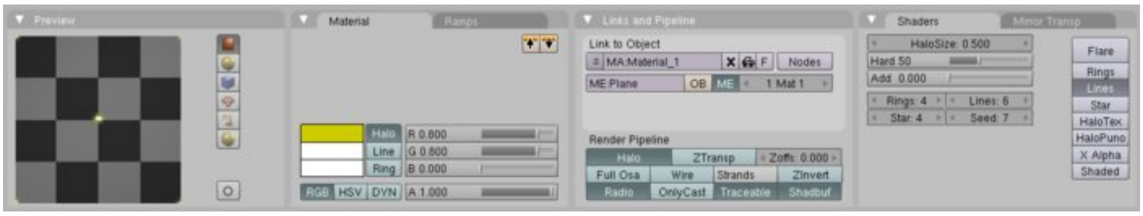
A group of buttons we have not used so far is the Children group.

Prepare and add a particle system to the plane. Adjust the parameters so that you get some particles flying into the sky, then increase the value of Prob: to 1.0. This will cause 100% of the particles to generate child particles when their life ends. Right now, every particle will generate four children. So we'll need to increase the Num: value to about 90 (*Children buttons*). You should now see a convincing firework made from particles, when you preview the animation with Alt A.



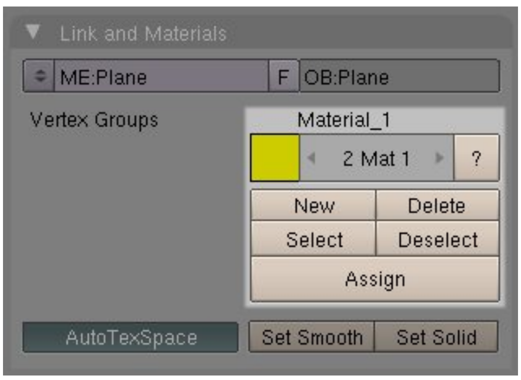
Children buttons.

When you render the firework it will not look very impressive. This is because of the standard halo material that Blender assigns. Consequently, the next step is to assign a better material. Ensure that you have the emitter selected and go to the Shading Context and Material Buttons (F5). Add a new material with the Menu Button, and set the type to Halo.



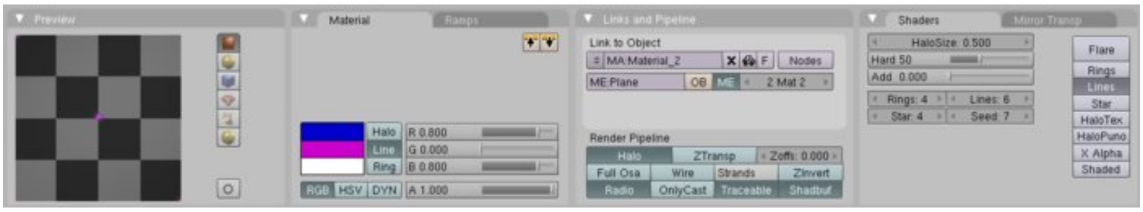
Firework Material 1.

I have used a pretty straightforward halo material; you can see the parameters in *Firework Material 1*. The rendered animation will now look much better, yet there is still something we can do. While the emitter is selected go to the Editing Context F9 and add a new material index by clicking on the New button in the Link and Materials Panel (*Adding a second material to the emitter.*).



Adding a second material to the emitter.

Now switch back to the Shading Context. You will see that the material data browse has changed colour to blue. The button labeled 2 indicates that this material is used by two users. Now click on the 2 button and confirm the popup. Rename the Material to "Material 2" and change the colour of the halo and the lines (*Material 2*).



Material 2.

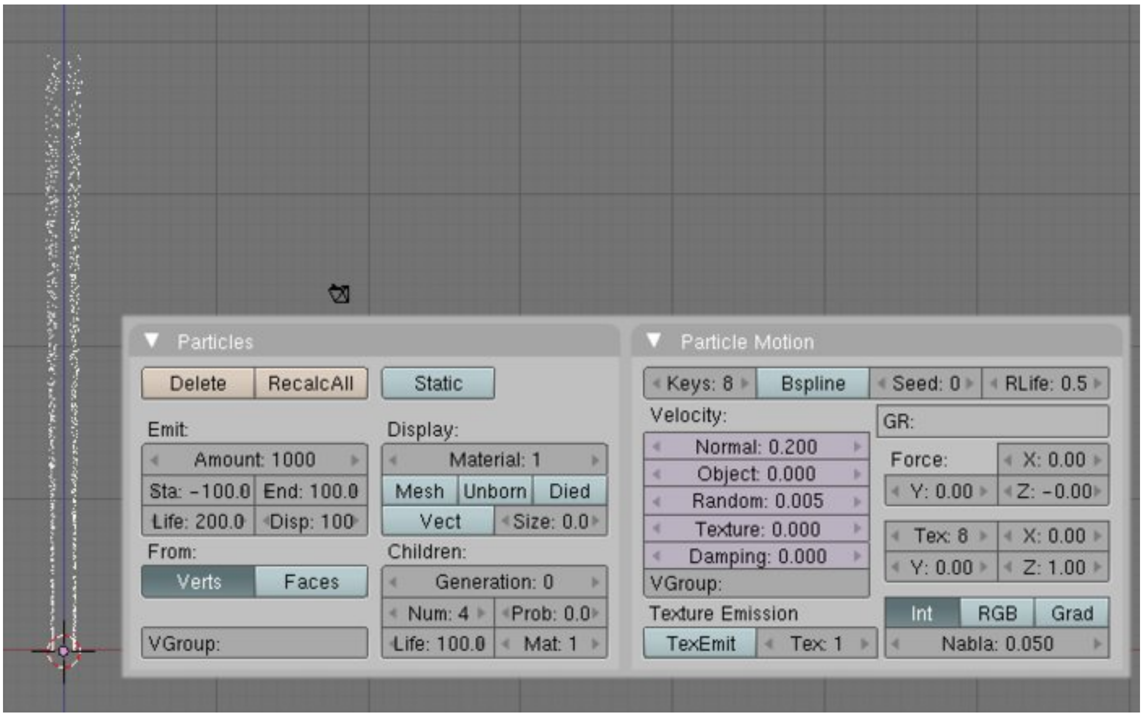
Switch to the particle parameters and change the Mat: button from the Children group to 2. Render again and you see that the first generation of particles is now using the first material and the second generation the second material! This way you can have up to 16 (that's the maximum number of material indices) materials for particles.

Further enhancements

Beside changing materials you also can use the material IPOs to animate material settings of each different material.

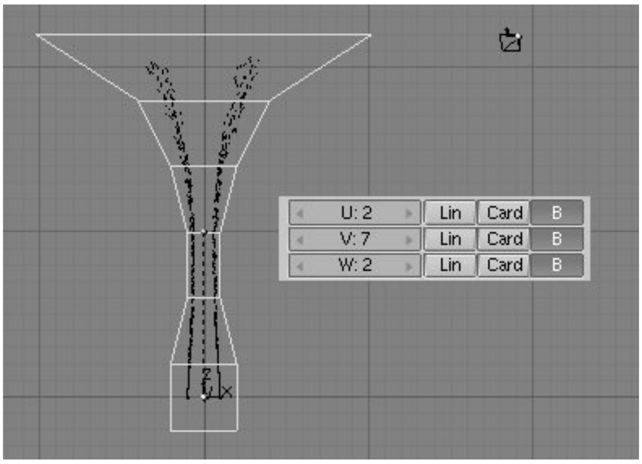
Controlling Particles via a Lattice

Blender's particle system is extremely powerful, and the course of particles can not only be determined via forces but channeled by a lattice. Prepare a single square mesh and add a particle system to it with a negative z−force and the general parameters in *Particle settings*.



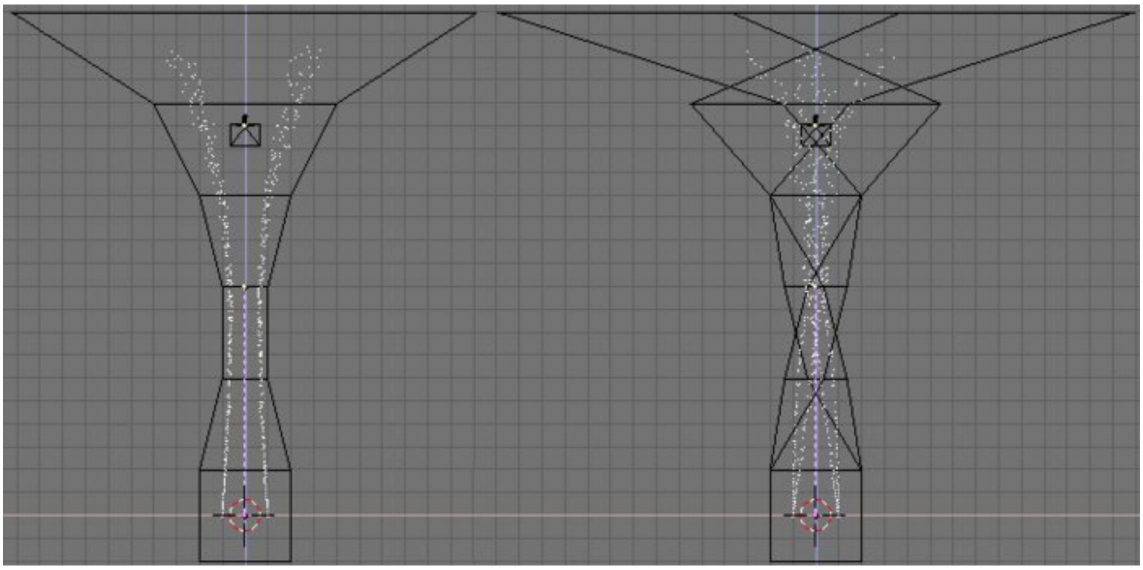
Particle settings.

This could be good for the smoke of four small fires in a windless day, but we want to twist it! Add a lattice and deform it as in *Lattice settings*.



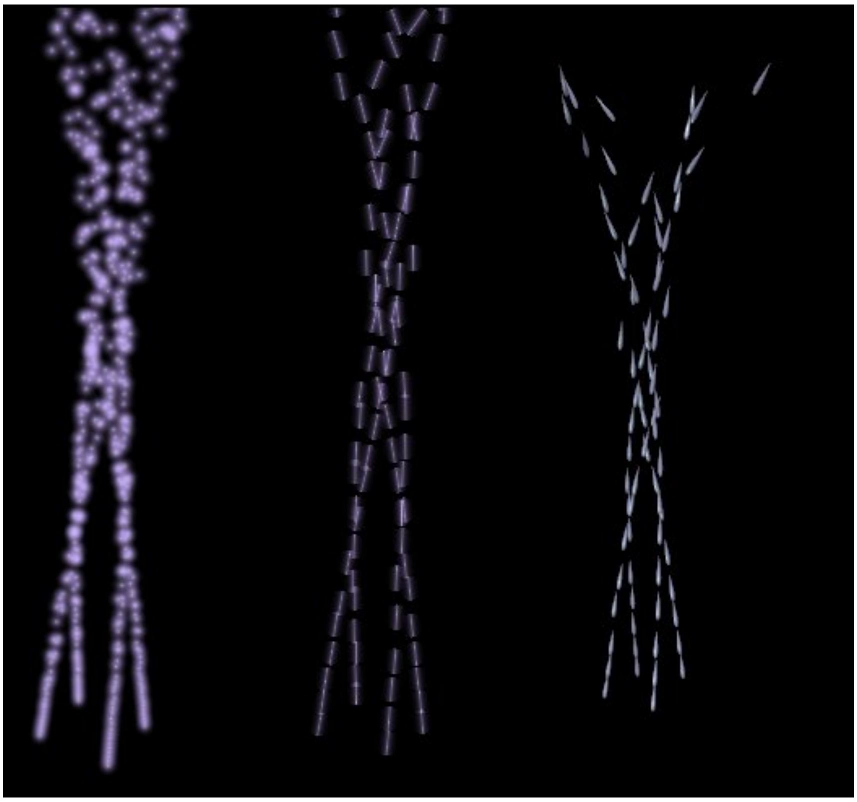
Lattice settings

Add a Lattice Modifier to the particle emitter and set the Lattice Modifier's **Ob:** to be the Object Name of the Lattice. If you now select the particle emitter, switch to the Object context (F7), Physics buttons, and press RecalcAll (note: in fact, the recalcl will usually be automatic), you will notice that the particles follows, more or less, the lattice (*Lattice deformation effects* on the left). As a further tweak, rotate each horizontal section of the lattice 60 degrees clockwise in top view, incrementally, as if you were making a screw. After this, recalculate again the particles. The result is in *Lattice deformation effects* on the right.



Lattice deformation effects

The twist is evident, and of course you can achieve even stronger effects by rotating the lattice more or by using a lattice with more subdivisions. If you give the emitter a halo material and you render you will see something like *Normal particles, left; Vector particles, centre; and DupliVerted objects following the particles, right.* on the left.



Normal particles, left; Vector particles, centre; and DupliVerted objects following the particles, right.

If you select the emitter, turn to Physics buttons and press the Vect button from the Particles tab, the particles will turn from points to segments, with a length and a direction proportional to the particle velocity. A rendering now will give the result of *Normal particles, left; Vector particles, centre; and DupliVerted objects*

following the particles, right. in the middle. If you now Duplivert an object to the emitter, by parenting it and by pressing the Duplivert button (Object buttons}}), the DupliVerted objects will have the same orientation of the original object if the particles are normal particles, but will be rotated and aligned to the particle direction if the Particles are set to Vect. By selecting the Original Object and by playing with the Track buttons you can change orientation (*Normal particles, left; Vector particles, centre; and DupliVerted objects following the particles, right.* on the right).

See Also

- BSoD/Particles − Blender Summer of Documentation article on Particles.
- Particles in Blender 2.46+ − Almost complete documentation for the new particle system in the Blender Wiki Sandbox section.

Previous: Manual/Effects and Physical Simulation	Contents	*Next:* Manual/Static Particles (Hair)

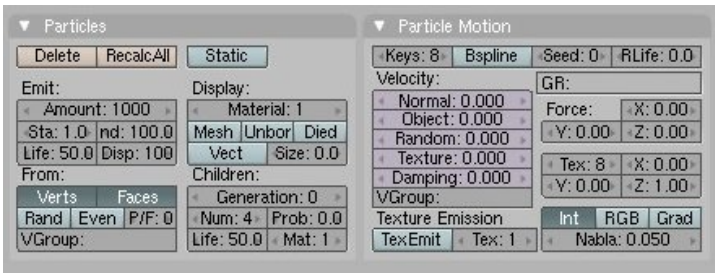
User Manual: Contents | Guidelines | Blender Version **2.40**

Blender Hair/Fur & Feathers

Overview

has an awesome particle generator that is not only capable of generating particles, but is also capable of creating hair−strands that can be used to create either hair, fur, or even feathers. This is achieved by converting the particles to renderable polygon strands. The polygon strands never go below 1 pixel in size, and this helps keeping rendering artifacts at bay when the rendering engine renders the strands.

Let us begin by taking a look at the particle generator and how it works. This is how the particle generator looks in Blender 2.44:



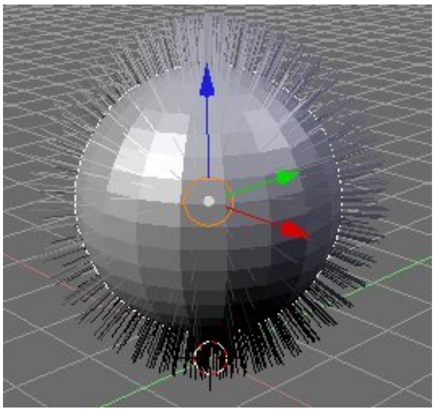
The particle generator

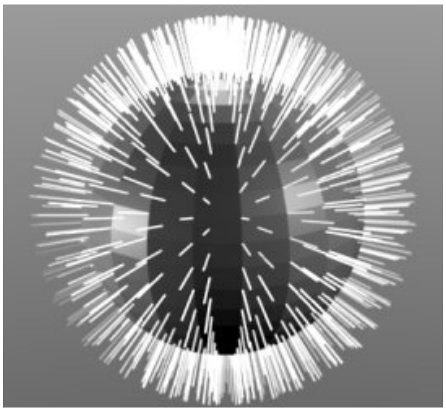
As you can see in *The particle generator* you have many options. To find this particle menu you must first create a particle emitter. The particle emitter must be an object of type Mesh. Let's just start by adding a sphere: Space Add Mesh UVsphere. Give the sphere around 20 segments and 20 lines. Exit to object mode (TAB).

Now go to your menu section and press F7 to get to the object menu. The object menu has two option buttons: Object ( ) or Physics ( ). Click the Physics button. There you will find a tab called Particles. Press NEW and you will see the menu just as in *fig−1* (except that by default, the two tabs are stacked).

Since we're about to create hair−strands or fur we need to make the particles static. Press static followed by animated if you plan to animate these later on. To draw strands between each particle generated we need to activate vector under the display section − press Vect. If you want the mesh to show during the generation of particle strands you simply press Mesh under the display section.

We need to see the particles now but we haven't informed our generator how to emit these from our mesh−object. To see these − go to the Particle Motion tab and set normal to `0.010`. Your sphere should look something like in *fig−2*. If not, then press the Z key to enter shaded view.


fig−2


fig−2b

They look quite boring don't they? That's because they're just a bunch of strands sticking out of a round ball, you can of course add these to any object you please.

Emitting strands From the mesh

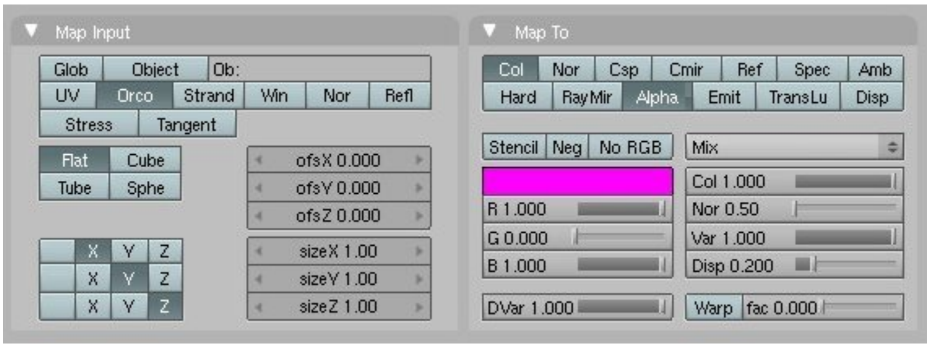
The mesh Emits particles. These particles are like photons; a certain amount of them are emitted, and they have a speed, direction, and life time. The more Amount that are emitted, the denser the hair will be. The longer these particles live ((have a height Life:)}, the longer the hair will be. These particles are packed very close together; many can fit inside a single pixel. It is a wasted of CPU time to calculate redundant particle locations, so set Step: at a low value ((`5` is the default).In the 3D views, you do not need to display/calculate 100% of the particles: a lesser percentage in Disp: will speed up your display.

You can also specify where to emit strands From: the entire object, or from just a certain portion of the object. There are two selections here in this section of the panel: Verts (vertices) and Faces. Select one or both as you wish. Verts emits strands only from the mesh's vertices. By default, all verticies emit strands. To emit strands only from a portion of the mesh, define a vertex group (see *Editing*(F9)) and type the name of the group in the VGroup input box (case sensitive). For example, if you had a human head, you would define a vertex group of `Scalp` and then enter that here. If you want to also emit hair from the spaces between the vertices (the faces of the mesh) select Faces. You will then get to choose between Random and Even distribution of hair/particles. For more natural/organic hair, choose Random. Dithering is a process of overlaying dots on top of one another to blur and soften. Choose a different dither to suit your hair style.

Let's make the strands better looking! To do this we will need to make the strands ease out and anti−alias into the environment around the object. We can even change the size of the hair strands and how the tip will look like, but more about that later, let's take a look at how we map the strands to look real soft and fluffy.

Strand Shader

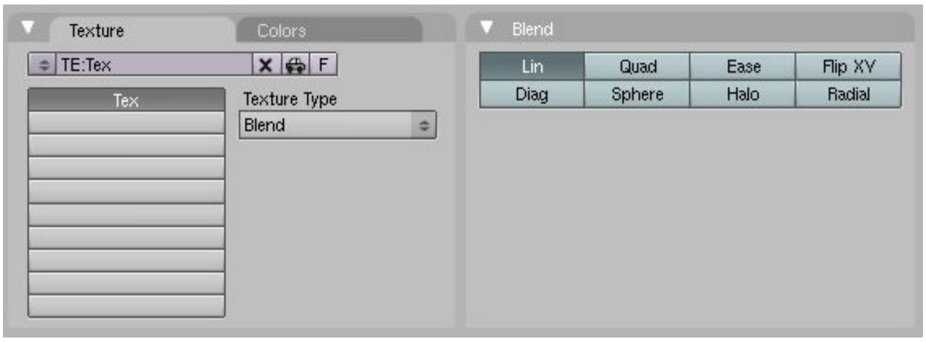
The strand shader has been implemented to follow the direction of the new strand polygons. In other words if you select strand and place a texture or a gradient − it will follow the direction of the strands. And we're going to use gradients to shade the strands so they fade out to nothing − giving us an illusion of a much higher resolution, this is known as anti−aliasing. You can't really render smaller than one pixel anyway − so we use this "trick" to make the hair−strand tips smaller and fade into oblivion.



fig−3

As you can see from *fig−3* you're now in the material menu (F5). From there you need to create a new material and add a texture, actually it would be practical if you create 2 materials if you intend to keep the original mesh−object as well as the particle strands, but for now we will concentrate on shading the particles. Please make your selection exactly as shown in *fig−3*. Strand activates strand mapping and Alpha activates the alpha channel. The alpha channel contributes to making textures opaque and the texture background invisible. This is the trick we discussed earlier to make the texture fade out to nothing.

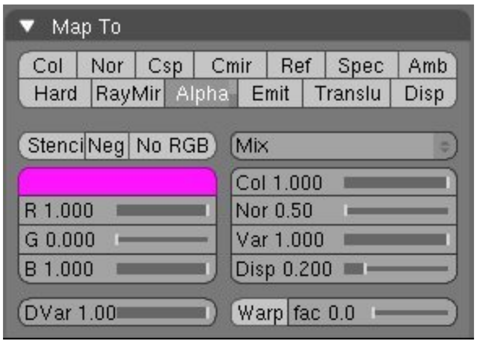
Let's go work on the "fading texture". We're going to add a Blend gradient to our texture slot−1. Check out *fig−4*.



fig−4

Now we're going to add color gradients to our blend−gradient texture, press Colorband, in the Colors tab. Change the default color of the first line marker to white with an alpha of *1.0* by LMB −clicking on it and moving all the color sliders to *1.0* and the **A**lpha slider to *1.0*. Make the second line marker black in the same way, this time making the alpha *0.0*. Make sure that you have the Alpha button selected in the Preview panel. If you've done everything correctly − things should look as in *fig−5*.

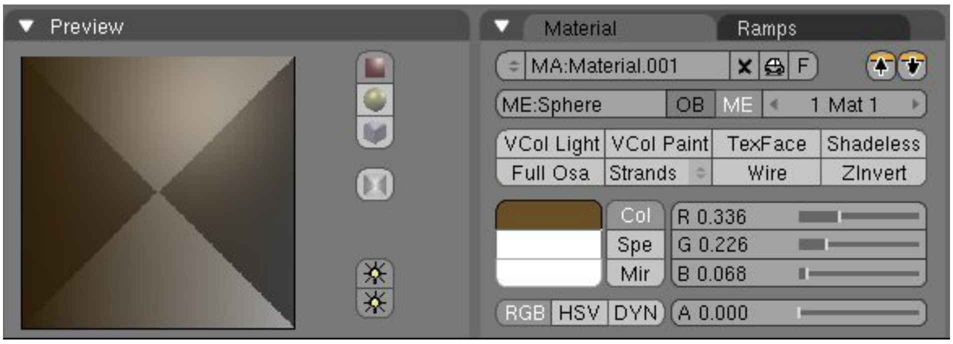
fig−5



fig−6

Ok, now that we've made the gradients ready to help create our fade effect on the particle strands, go back to (F5) the material menu and untick the COL button from the Map To tab (*fig−6*). The reason we do this is to use our own colors instead of the color gradient we made earlier. We just wanted to use that as a fade gradient from 100 % opaque to 0.

Now we're going to make a "hair" color, just go to the material menu and adjust the colors as you see fit and slide the Alpha slider (A) to zero. We do this to make the alpha channel gradient we made in *fig−4* control our alpha 100%. You can of course adjust this here to your liking if you don't want 100% alpha, but do this for now.

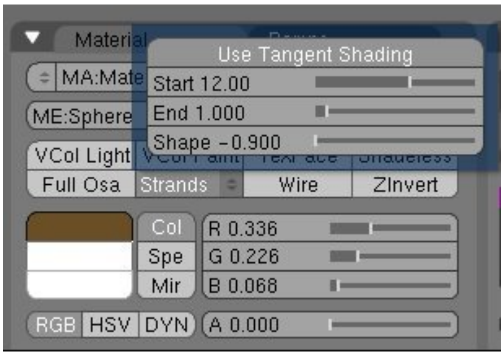


fig−7

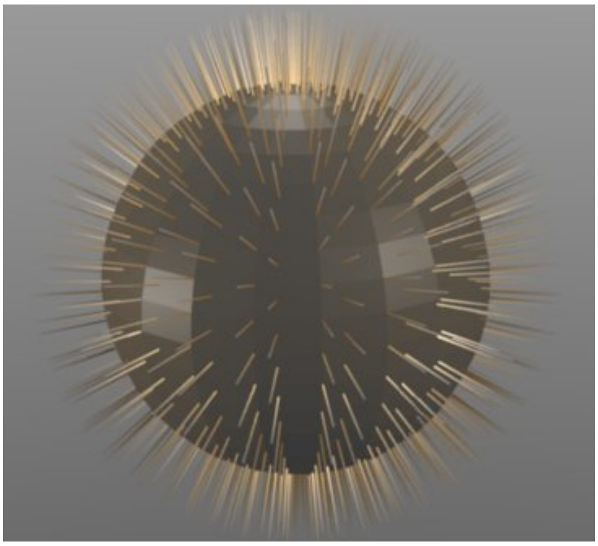
If you did everything as described above and press F12 to render, your render should look something like *fig−8*.

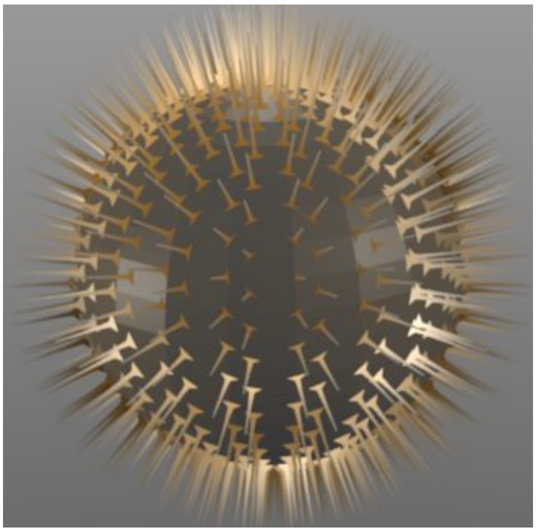
As you can see − we now have soft "fluffy" hair strands, but we can do even more yet before we add more particles. Blender has yet another option under the material menu, Links and Pipeline tab, Strands button, called tangent−shading. The tangent−shader creates a thicker/thinner hair/fur−strand curve depending on the length of your particle strands. You can adjust it to make the strands start out or end thicker/thinner and how sharp/rounded they will be.

I've set the shape to `−0.900` (*fig−9*) so it's really "spikey".



fig−9


fig−8



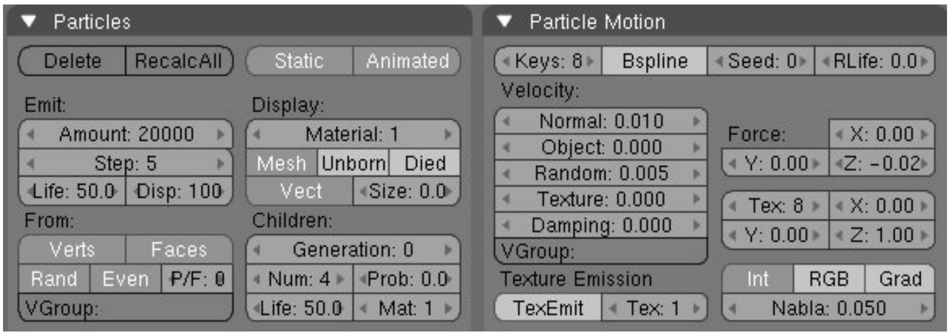
fig−10

And if you render it, *fig−10* shows how it would look.

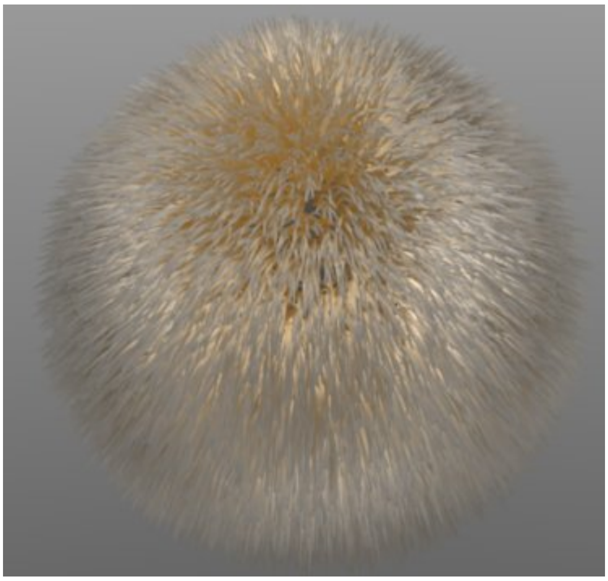
Now that you've done all the footwork you're ready to have a LOT of fun with hair/fur, let's go back to the particle editor and create a lot of fur.

In your particle menu − let's add a lot more particles...say `20000` of them, if you don't have a very strong computer you can change the DISP value to display 20 percent or so and when you play with particles, only 20 percent of them will show up in your viewport but at rendertime − all particles will be present.

Now that we have a lot of particle strands let's add some weight to it. As we know...gravity goes downwards so we change the z−force value to about `−0.02` and to make our fur a bit frizzy and random looking − add `0.005` to the Random widget under the Velocity: section. To create an even better looking distribution of our fur we could get our particles to emit from vertices (Verts) and Faces both randomly and evenly, you can set this as in *fig−11* under from.



fig−11

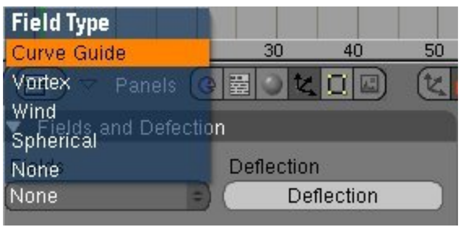


fig−12

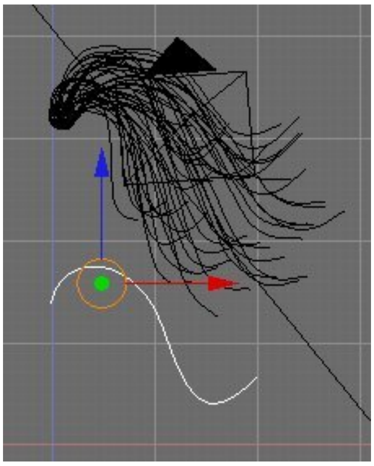
If you render this, it will look something like in *fig−12*. If it doesn't, then it's possible that your light setup is very different from the one used in the example. Lights are very important to particle−strands, obviously because of shading. If you need to do proper lighting then check out the lighting tutorials or lights section in the manual and do this tutorial over again, you can get awesome results if you do your homework when it comes to proper lighting.

Let's make hair with hairguides

That's remarkably easy to do with the new guides, all you have to do in order to guide the hairs in your new hairball is to add a curve and tell it it's a curve guide. To do that Space Add Curve Bezier Curve and with the curve selected go to your F7 menu and take a look in the dropdown menu found under the fields and deflection menu:



fig−13b



fig−13

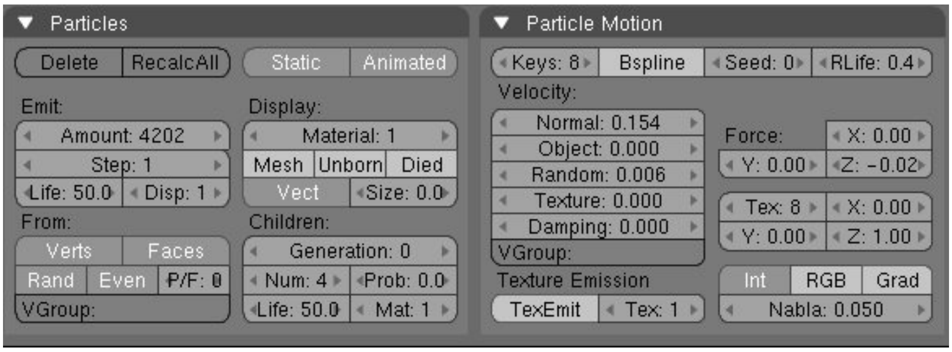
Now that you have selected it to become a curve guide you will see that it instantly affects your particle furball. As you also might see − the shape of the hair is now conformed to the guide. You're of course free to shape the curve any way you see fit, Blender will real−time update when you move the bezier curve segments around and add new segments with E for extrude.

Again I cannot stress enough how important lights and shadows are to get quality hair, learn to do your light setup well and you will be rewarded with lush and beautiful looking hair. A good way to get shiny clean hair is to reduce specularity just a tad and increase the hardness value to about 70−100.

A thing you might want to do in order to render more lush "fluffy" hair is to select the Ztransparency (ZTransp) option found in the Material buttons (F5), Links and Pipeline tab. This will render the strands in alpha layers with transparency making it antialias even better and give you totally awesome results, of course...this WILL cost you render time but it's well worth it. You can use Ray−Transparency (Ray Transp, Mirror Transp tab) and set depth to whatever you see fit − and it will render somewhat faster, but it won't look nearly as good as with Ztransparency.

fig−14

Nice hair huh? Want to see the settings for it? Here you go, check out *fig−15*.

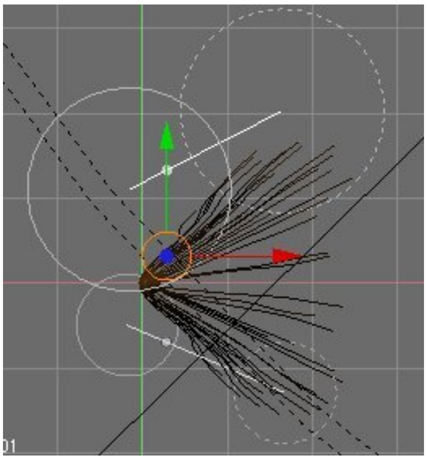


fig−15

Let's take a look at what we've done to get that nice hair. I made the previous object a bit smaller by scaling it so the start−emitter of the hair would be smaller (just for effect − really). Then I blew up the Normal (speed) value found under the Particle Motion section (Velocity:), gave it a little random−life (`0.4`) to vary the speed a bit. I also scaled and moved around the bezier curve guide to affect the hair try these variations − you'll be surprised how flexible it can be and how many variations you can make.

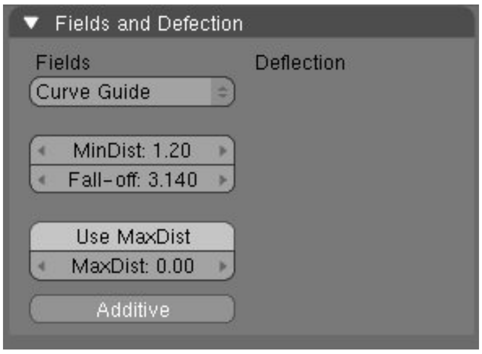
Want to get even more advanced? Read on...

Advanced hair



fig−16

You can use multiple curves to guide your hair around. Make a duplicate of your curve and tilt them both slightly as shown on *fig−16*.



fig−17

Now check out the F7 menu − fields and deflections again and press the Additive button. This button creates an approximation field with your curve guides so when you blend two or more together − it will affect parts of your hair−strands depending on your settings in MinDist and Fall−off. Make a small MinDist value such as shown in *fig−17* for both curves and experiment with the Fall−off values (small numbers to start with). If you now mix these two curves together placing them close to the hair emitter you can see the effect real−time and adjust the Fall−off/MinDist/MaxDist values accordingly. These values will depend on your curves, number of segments and emitter size etc, so in other words they will vary with your own creations.

If you've done everything correctly − you should be able to control your hair splitting it into sections such as shown here:

fig−18

As you can see from *fig−18* you now have full control over where you put your hair. You can even vary the length of each "curve" so it will affect the length of the hair that are near the curve.

It doesn't end here − not by a long shot! There's nearly no end to how much you can control your hair with Blender. You can twist it around, knit it, swirl it, toss it and yes do anything you want with it. Become a world famous hairdresser − your characters will never go bald again.

Tommy Helgevold (JoOngle)

Hair tutorials

- Video tutorial on how to make hair
- Tutorial on CGTalk

See Also

- Hair particles in Blender 2.46+ − Mostly incomplete documentation for the new hair particle system in the Blender Wiki Sandbox section.

Previous: Manual/Particles	Contents	*Next:* Manual/Soft Bodies

User Manual: Contents | Guidelines | Blender Version **2.46**

A Softbody is really a physical simulation of what would happen "in the real world" if the mesh actually had substance and was a real "thing". Using Softbodies, you can simulate the shapes that a mesh would take on if it was real and have volume, was filled with something inside of it, and was acted on by real forces.

What's the difference between a bar of metal, a bar of rubber, and a bar of gelatin? Each will bend under the force of gravity if you hold it out by one end, but by very different amounts. To model a bending object, you could carefully adjust all of the vertices yourself, or you could try to insert some kind of bony armature inside the object. In the end though, you are only guessing what would really happen. Blender provides a much simpler alternative called "soft bodies" simulation. If you add the "soft body" modifier on a mesh, Blender will compute the interaction between the mesh and various environmental forces (like wind or gravity), and Blender will distort the mesh accordingly.

Softbody calculation is like an automatic animation. While you control what the world will do to the Softbody, and where it is pinned, Blender really takes over and calculates what the shape of the mesh will actually be at any given time. That means that it might bounce around in your scene, if that is what would happen in the real world based on your virtual reality. Therefore Softbodies are affected by:

- Deflections from hard bodies (meshes)
- Deflections from other Softbodies (e.g. two blobs of jello colliding)
- Gravity
- Fields (e.g. wind, vortex, spherical repulsion/attaction)
- Itself (e.g. a piece of cloth, folding over itself, stops itself from folding)

Only Mesh objects can be a Softbody.

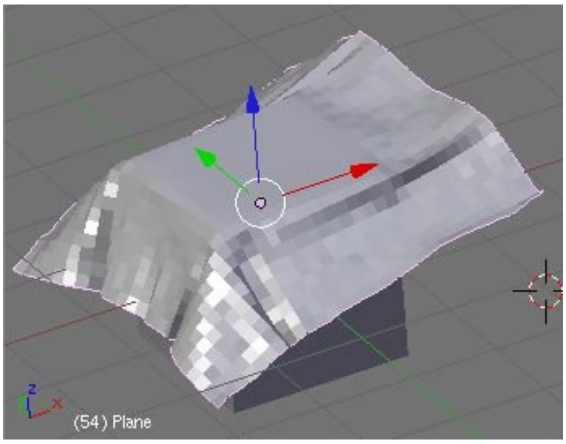
Soft Bodies

Mode: Object Mode

Panel: Object context Physics sub−context Soft Body and Editing Context Modifiers

Hotkey: F7 to get to Object context; repeat to change sub−context.

Description



Tablecloth simulation

In between each neighboring vertex of a mesh, you typically create edges to connect them. Imagine each edge is really a spring. Any mechanical spring is able to stretch under tension, and able to squeeze under pressure. All springs have an ideal length, and a stiffness that limits how far you can stretch or squeeze the spring.

In Blender's case, the ideal length is the original edge length which you designed as a part of your mesh, even before you enable the "soft body" modifier. Until you add the modifier, all springs are assumed to be perfectly stiff: no stretch and no squeeze.

Once you add the Soft Body modifier, you can adjust the stiffness of all those edge springs, allowing your mesh to sag, to bend, to flutter in the breeze, or to puddle up on the ground.

There are two main methods to control the soft body effect:

Goal
: Soft body Goal acts like a pin on a chosen set of vertices; controlling how much of an effect soft body has on them. With Goal fully active (1.0), the object will act like any regular animated object (no soft body effect). When setting Goal to 0.0, the object is only influenced by physical laws. By setting Goal values between 0.0 and 1.0, you can blend between having the object affected only by the animation system, and having the object affected only by the soft body effect. Goal also serves as a memory, to make sure soft objects don't deform too much, ending up in the non−soft animated shape. Using the Vertex Group weight system, you can define a Goal weight per vertex. To make this look more natural, spring forces can be defined to control how far vertices can move from their original position.

Springs
: The Edge Spring Stiffness defines how much edges try to keep their original sizes. For example, by adding diagonal edges within a cube, it will become stiffer (less "jelly like"). By tweaking the E Stiff parameter, objects can be set to try to, more or less, keep their original shape, but still move freely with dynamics.

Softbody vertices interact with all the forces applied (usually to particles) in the layer, such as wind, force fields .. and what ever Physics Field effect is on a common layer.

Softbodies are also pushed by Fluids, as long as that fluid domain is marked as a deflector. Also, the Softbody has to be within the fluid's domain.

Softbodies have materials and textures just like any other mesh, and can be subsurfed, multires, and have other modifiers. In fact, when you designate a mesh to be a Softbody, you may notice the Softbody modifier in the modifier stack. When used with other modifiers, like the Subsurf modifier, the Softbody must be the first, or top, of the stack, since it operates on the base mesh. In fact, the Subsurf modifier is an excellent way to smooth out a Softbody without adding more vertices (and thus slowing down computation). A Softbody can have shape keys as well.

Interaction with Deflector

The simulation is really about an interaction between two objects: the Softbody and a Deflector (which can be a hardbody mesh, or another softbody). The deflector settings work in conjunction with the Softbody settings to provide accurate results. So, if you are not getting good results, be sure to check both objects.

In particular, both objects have Damping settings, but from their individual perspective. A couch cushion, a soccer (football) field and a basketball court are all Deflectors. A couch cushion that is very plush will absorb all of the force of a Softbody sitting on it; it therefore has a dampening of 1.0. A soccer field is turf and has much more cush than the hard surface of a pine wood floor. Therefore, the soccer field will dampen any softbody that it interacts with, perhaps by 0.5 depending on the length and strength of the grass, how hard the

dirt is packed, and how much moisture it contains. It will absorb some of the blow, but not all. A hardwood floor, on the other hand, has 0.0 dampening effect.

When a ball bounces off the hardwood floor, it bounces a second time less, and less, until it eventually comes to rest. Therefore, if the floor didn't stop it, something about itself must dampen the spring interaction, or else it would keep bouncing forever. Therefore, the Softbody itself must have some amount of Edge dampening.

The similar argument goes for Error Limit; the Deflector has an inner/outer edge buffer zone, and the Softbody does as well. Adjust both in tandem to get acceptable results.

Workflow

Defining and using a Softbody is a quasi−animation task. However, important considerations before animation fit into and augment the normal workflow:

- Concept phase
  - Designate which objects are to be Softbody
  - Define how they are to jiggle and move, what gravity is
- Storyboarding
  - Annotate physical simulated forces (wind, vortex)
  - Annotate the action of deflectors via arrows
  - Pick camera angles that hide possible collisions
- Modeling
  - Use low−poly modeling on Softbody objects (or at least, lower, depending on CPU speed)
  - Apply Subsurf modifiers
- Lighting
- Animation
  - Apply Softbody modifier and begin Goal and weight painting
  - Run simulations

Options

Softbody settings.

Soft Body
: This enables the soft body modifier on the selected object

Automatic Calculations
: With versions prior to 2.46, you had to bake, or pre−compute the deformation of the mesh. This approach presented problems when using a render farm, or when you changed parameters and forgot to re−bake. With version 2.46 onward, the simulation computation happens automatically as needed.

A cache, actually a set of disk−based files in a directory, saves the mesh shapes. These files are automatically created for you and saved when you save the .blend file. Using a cache eliminates a separate Bake step in the workflow, making the simulation process much more interactive and removes an interruption in the workflow.

Protect

You can *Protect* the cache from Softbody parameter changes. Enable this when you have them set, to avoid unwanted changes.

Clear

If you make changes to the Softbody or to objects that it collides with, you should *Clear* the unprotected cache and force Blender to recompute the Softbody shapes when they are needed. Otherwise, the Softbody *may* not pick up on the changes and will give you the cached results. An example of this is when you have changed the animation path of a colliding object; you should select the Softbody that it collides with (or used to collide with) and clear its cache.

Cache:

You must save your .blend file, so the solver knows where to put the cached results. When you first save your blend file, a //blendcache_softbody\ folder is created to hold the cached mesh shape information. If you move the .blend file, you should move these files as well, or Blender will have to recompute them in the new location. A 500−vertex softbody, computed for 250 frames, takes 1Meg of disk space. If you have never saved your .blend file, Blender has no place to put the files. Also, be sure Blender has security rights on your OS to create folders and files.

Friction

Sets the amount of friction the object 'feels' against the void/gas/liquid surrounding. 50 is no slipping at all, 1.0 is is like silk and can easily slide. Even the weight of the softbody itself can make a slippery cloth slide off an object.

Grav

Gravity, the amount of force in Z−axis direction. Earth gravity is a value close to 9.8 m/s2. Positive values make Softbodies drop; negative values make them rise.

Mass

Mass value for vertices. Larger mass slows down motion, except for gravity where the motion is constant regardless of mass. May be it's time to dust off the books on physics (yah that ol' school book that 'accidientially' dropped behind the 'hitchhiker') .. reading Newton's laws of motion. The mesh is assumed to have even density

Speed

You can control the internal timing of the Softbody system with this value. 1.0 uses normal mass, friction, and gravity. Use values less than one to simulate dense air or very light fabric dropping, or very heavy or stiff fabric in the breeze.

Use Goal

Enabling this tells Blender to use the motion from animations in the simulation (Ipo, Deform, Parents, etc). The "Goal" is the desired end−position for vertices based on this animation. How a softbody tries to achieve this goal can be defined using stiffness forces and damping.

Goal

The example image has a vertex group "Hands" which are two clumps of vertices along the edge of the mesh, simulating where two hands would be holding it and forcing it down. These vertices travel more closely with the object's animation and thus impart force to the mesh's movement.
If no vertex group is used, this numeric field is the default goal weight for all vertices when no Vertex Group is assigned. If a vertex group is present and assigned, this button instead shows an popup selector button that allows you to choose the name of the goal vertex group.

G Stiff

The spring stiffness for Goal. A low value creates very weak springs (more flexible "attatchement" to the goal), a high value creates a strong spring (a stiffer "attatchment" to the goal).

G Damp

The friction for Goal. Higher values dampen the effect of the goal on the soft body.

G Min/GMax
: When you paint the values in vertex−groups (using WeightPaint mode), you can use the GMin and Gmax to fine−tune (clamp) the weight values. The lowest vertex−weight (blue) will become GMin, the highest value (red) becomes GMax. (please note that the blue−red color scale may be altered by User Preferences)

Use Edges
: The edges in a Mesh Object (if there are, check Editing−>Mesh Panel) can act as springs as well, like threads in fabric.

Stiff Quads
: For quad faces, the diagonal edges are used as springs. This prevents quad faces from collapsing completely. Enabling this can make a sheet of cloth behave like a thin sheet of metal or thick piece of rubber.

CEdge
: Checks for edges of the softbody mesh colliding

CFace
: Checks for any portion of the face of the softbody mesh colliding (compute intensive!) While Cfaces enabled is great, and solves lots of collision errors, there doesn't seem to be any dampening settings for it, so parts of the s−b object near a collision mesh tend to "jitter" as they bounce off and fall back, even when there's no motion of any meshes. Edge collision has dampening, so that can be controlled, but Deflection dampening value on a collision object doesn't seem to affect the face collision.

E Pull
: The spring stiffness for edges (how much the edges are stretched). A low value means very weak springs (a very elastic material), a high value is a strong spring (a stiffer material) that resists being pulled apart. 0.5 is latex, 0.9 is like a sweater, 0.999 is a highly−starched napkin or leather.

E Push
: How much the Softbody resist being scrunched together, like a compression spring. Low values for fabric, high values for inflated objects and stiff material.

E Damp
: The friction for edge springs. High values (max of 50) dampen the E Stiff effect and calm down the cloth.

Aero
: Enable the N button if you want to use an aerodynamic model that is closer to physical laws and looks more interesting. Disable for a more muted simulation
: Edges can feel wind as they move, and sail or flutter in the breeze. A simple aerodynamic model of a flag sailing in the wind. Nice value approx. 30. You must use a separate mesh that generates the Wind Field (physics buttons), or animate (move) the softbody in 3D space, which is assumed to be filled with air.
: Technically, a force perpendicular to the edge is applied. The force scales with the projection of the relative speed on the edge (dot product). Note that the force is the same if wind is blowing or if you drag the edge through the air with the same speed.
: That means that an edge moving in its own direction feels no force, and an edge moving perpendicular to its own direction feels maximum force.
: In between you have the forces like they are when you fly a kite.

Bend
: How much starch you want, Charlie? More rigidity and the Softbody is stiffer across a wider area (number of vertices). For example, new denim is much more rigid than silk. A steel bar is much more stiff than a steel tube.

Shear
: How much force it takes until it suddenly gives way and bends. A spring, or copper tube has a much lower shear value than a steel tube.

Soft Body Solver

Mode: Object Mode

Panel: Object Context Physics SubContext Soft Body Collision tab

Description

The next step is to tell Blender how to use that basic information when performing the simulation. A "Solver" is an approach and settings that Blender uses when performing the simulation. Error settings control the fineness of the algorithm.

Options

Image:Manual−Softbody−Solver.jpg

Solver Select
: Blender has two solver algorithms that it can use:

- Soft − a step−size, adaptive algorithm for most situations
- Runge Kutta Correct Physics − a mathematically correct and educationally correct algorithm. Tends to be "unstable".

Error Limit
: Rules the overall quality of the solution delivered. Default 0.1. The most critical setting that says how precise the solver should check for collisions. Start with a value that is 1/2 the average edge length. If there are visible errors, jitter, or over−exaggerated responses, decrease the value. The solver keeps track of how 'bad' it is doing and the 'error limit ' causes the solver to do some 'adaptive step sizing'.

Velocity Check
: Enable to help the Solver figure out how much work it needs to do based on how fast things are moving.

These settings allow you to control how Blender will react (deform) the soft body once it either gets close to or actually intersects (cuts into) another collision object on the same layer.

Choke
: Default 0. Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Fuzzy
: Default 1. The very last chance to get really fast collision situations integrated in limited time. This number is multiplied to the Error Limit when vertices are detected to be inside the collision mesh. Trades off with the stability/compute time of the simulation.

M button
: If you turn on the Monitor button you'll get a print on the console how the solver is doing.

MinS − Minimum frame step
: Default 10. To avoid misses on collision, the minimal step size MinS should be something like 10 or more. The positions of the (collision) objects then are (at a minimum) interpolated in MinS sub − positions per frame. For example, if your frame rate is 25 fps, then a setting of 10 would mean that Blender would look for intersections 10 times a frame (every 0.004 seconds of real−time animation). Think of a cube that is at the right hand side of the wig in frame n and on the other side in frame n+1.

If you don't explore what happened in between you simply miss the collision.

MaxS

Default 0. MaxS is a kind of emergency brake to enforce really bad situations to obtain a result at all. Use a number that is much larger than MinS. It is the maximum number of steps per frame to be taken. It overrides the adaptive stepsize calulated form Error Limit if it gets too small. However the solution is less acurate then. A value of 0 disables this option (and Blender may spend a long time figuring out the deformations).

Speed vs. Stability

When setting up the simulation, there is a wide variety of uses for Softbody simulation under a wide variety of settings and situations. Therefore, there is no single right answer for these settings. If there was, we would automate them. In general, the simulation has competing goals:

- Compute time: you don't usually want to wait two days while Blender computes the exact placement of ever vertex, but you don't want it to be totally inaccurate either
- Collision Detection: If a fast moving object goes through a Softbody, even for a split second of a frame, you want to see some effect of that collision in a subsequent frame, even though the collider is long gone.
- Collision Resolution: If two meshes do collide and perhaps intersect, you want Blender to rebound the merge area quickly, but not so fast to to cause it to fly off into space
- Steady State: When the body comes to rest, you don't want to see it quiver, although in real life everything has a frequency and quivers somewhat

In general then, you want to run your simulation with the default settings, increasing Error Limit (and thus decreasing compute time) until you start to see issues.

Soft Body Collision

Mode: Object Mode

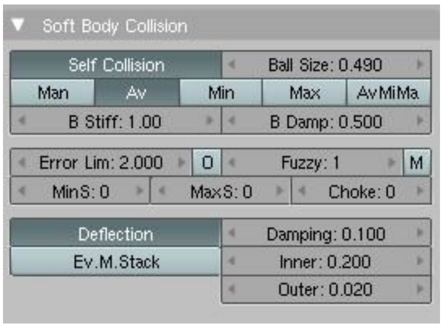
Panel: Object Context Physics SubContext Soft Body Collision tab

Description

The Soft Body is a mesh that deforms over time. That deformation is computed based on real−world physics. Part of that physics includes how the mesh will deform as it collides with 'solid' objects, including other soft bodies, on the same layer. This panel allows you to control how this cloth (soft−body) interacts with those other solid objects.

The reason for the layer restriction is for optimum use of computing resources. Softbody calculations, like Fluid calculations, can take a long time. So, usually, restrict your Softbody to a layer by itself, and then share that layer only with other objects that come in contact with it. To speed things up even more, if the Softbody only interacts with part of a mesh, duplicate only part of the mesh and move it to the layer.

Options



Collision settings

This panel has two parts:

- Self−collision − used to make sure the softbody does not intersect itself (think of a curtain or flag blowing in the wind)
- Deflection settings − reflect the Fields and Deflection panel settings; for convenience allows them to be set here as well.

Self Collision

When enabled, allows you to control how Blender will prevent the softbody from intersecting with itself.

Ball Size
: Default .49 BU or fraction of the length of attached edges. the edge length is computed based on the algorithm you choose. You know how when someone stands too close to you, and feel uncomfortable? We call that our "personal space", and this setting is the factor that is multiplied by the spring length. It is a spherical distance (radius) within which, if another vertex of the same mesh enters, the vertex starts to deflect in order to avoid a self−collision. Set this value to the fractional distance between vertices that you want them to have their own "space". Too high of a value will include too many vertices all the time and slow down the calculation. Too low of a level will let other vertices get too close and thus possibly intersect because there won't be enough time to slow them down.

Ball Size Calculation Algorithm
: Because of the infinite number of ways to model a softbody, and thus their geometric relationship to each other, Blender gives you 5 different algorithms to choose based on the shape of the softbody. By default, the Average algorithm is used and works like this: The average length of all edges attached to the vertex is calculated and the multiplied with the ball size setting. So for a regular mesh with ball size 0.49 the 'personal spaces' are calculated such that everyone is feeling fine but would not want to get any closer. <−− needs review .. how collision balls are calculated using automatics BM

B Stiff
: Default 1.0. How elastic that ball of personal space is. A high stiffness means that the vertex reacts immediately to someone invading their space, like an uptight girl on the bus.

B Damp
: Default 0.5. How the vertex reacts. A low value just slows down the vertex as it gets too close. A high value repulses it.

Deflection Settings

Use this panel to make this Softbody deflect other Softbodies. These settings are identical to those used for Hardbodies.

Damping

The amount of bounce that surfaces will have, ranging from:

◊ 0.0 − No damping, soft bodies will have maximum bounce, to
◊ 1.0 − Maximum damping, soft bodies will not bounce at all.

Inner / Outer

An artificial padding distance added to the inside and outside of each face, to help prevent intersections. Softbody will come to rest this distance away from the face of the colliding object. Adjust to prevent tears and the Hardbody poking through a face of the Softbody.

Deflecting Objects must be Real

Arrays, and modifiers, including mirrors and duplicates, must be applied and be 'real' mesh objects in order to be detected by the Softbody, and to react to them and be deformed by them.

Examples

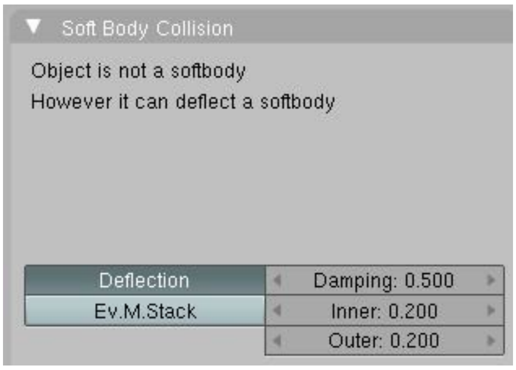
Many users have asked, what settings should I use to get useful results? This section suggests some settings that we know work. The issue is that Softbodies can be so many things:

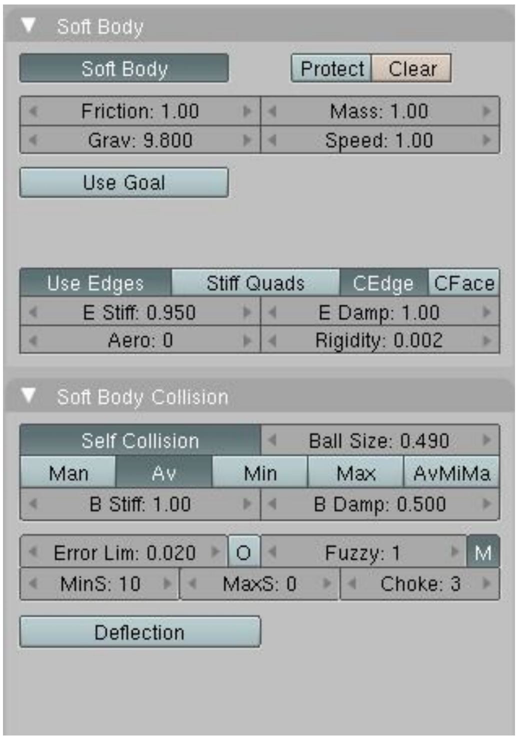
- Cloth and all its applications: tablecloths, flags, clothes
- Woven fabric, knitted
- Trampoline and all other woven elastics; diving boards
- Flesh, water balloons, inflated tires
- Stretchy shapes, Latex, Mylar
- Jello, Gelatin, sponge
- Rubber Ball, baseball, basketball, football, tennis anyone?
- Magnet, victims of Darth Vader
- Carpet with a thick under padding, pillows, cushions
- Wrestling or workout mat, boxing gloves, punching bags
- Foam and/or spring−supported objects, like chairs, couches, seats
- Toys, especially foam−based, like pool noodles, #1 fan gloves
- Balloons, air mattresses and other low−pressure air−filled objects
- Melting, leaking stuff (this is where Softbodies cross over fluids)
- Gazoober (whatever your imagination wants it to be)

Using a negative Gravity, you can simulate:

- Helium−filled balloons
- oil drops rising to the surface in an underwater scene
- air bubbles underwater
- hot air rising (geothermal tidal currents)
- hot air balloons

Tablecloth





In this suggestion, we use a Cube as a Deflection object that is 2 BU cubed. The Softbody itself is a plane that is 4 BU square, and subdivided 19 times, so that each face is 0.2BU square. This is important, since many Softbody settings are measured in Blender Units, and collision detection and deformation can only be done where there are edges. For the colliding object, (the cube that represents the table), set Softbody deflection as shown:

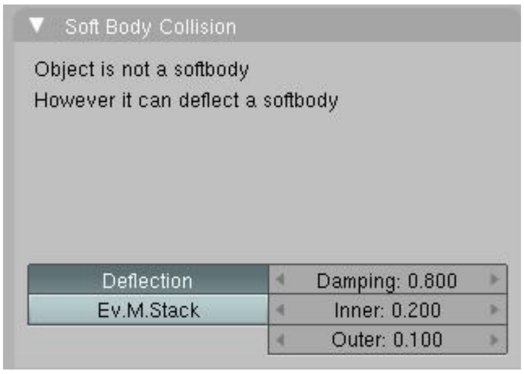
- Damping 0.5 (simulates some air between the table and cloth. 0.2 makes the cloth fly off the table, 1.0 sticks like glue)
- Inner: 0.2
- Outer 0.2 (0.1 yields corner cuts; use 0.2 for no cuts)

The key is the dampening; you want the surface not to excite or repel the Softbody, but not so high as to have it stick like a vacuum. You want an outer shell (collision detection area) big enough to prevent cuts. Too big of an shell and the Softbody will "float" above the object and appear to puff away from the sides like repelling magnets. Too small and corners will cut through the cloth. As an alternative, bevel the edges of the table; our simplistic cube provides a very sharp, harsh corner.

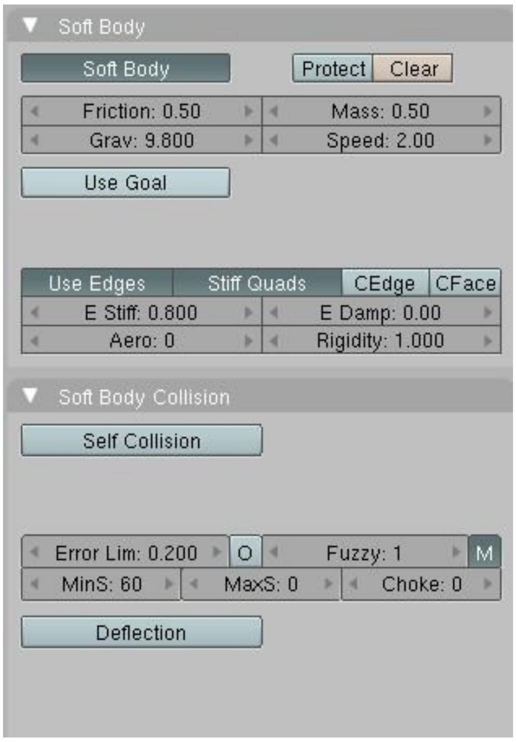
The Softbody settings for the tablecloth itself reflect a table on earth that is clean and waxed (*Friction*). It is a light material with a teeny bit of stretch (*E Stiff*), not cotton but with a little rayon for an alive feel (*E Damp*). It is used in a restaurant so it has some starch for formality (*Rigidity*). Edges have to be used for collision avoidance.

Enable self−collision, so that it does not fold through itself. The *Error Limit* is one−tenth the size of an edge. Monitoring allows you to watch the console − so exciting.

Soccer Ball or Football



The soccer field is a plane and is the Deflector with the settings shown to the right. The field has short grass (Damping) that allows the ball to roll and slide, but not alot.



For the ball, use an Icosphere with 2 Subdivisions and a Radius of 0.5. This is your softbody. It is fairly slippery (Friction), as the surface is polished/waxed leather. It is lightweight (Mass) and fast to react (Speed).

While the leather can give (E Stiff) when kicked, it quickly gets back into shape (E Damp). Wind does not deform it (Aeor) and it is very stiff (Rigidity) when dropped onto a hard surface.

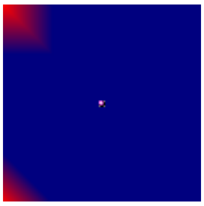
We don't worry about someone kicking the ball so hard as to blow it out (Self Collision). When it hits the ground, it kind of like plops down into the grass (Error Lim) and stays there (MinS).

Flag

This example will show you a way to do a simple flag moving with the wind. The steps are:

1. Create the Flag
2. Assign Weights to a Vertex Group
3. Designate it as Softbody
4. Add Wind

Create a plane in front view and subdivide it three times. Go to the Modifier panel activate *Subsurf*. In the Editing context Link panel, press Set Smooth.

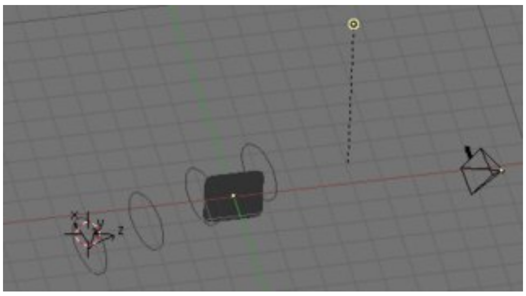


Example Weight Setting.

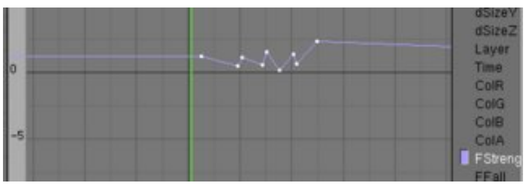
Add two pins to our flag in both upper and lower corner of our plane, simulating where the flag would be attached to the flagpole. Create a new Vertex Group, and set Weight to 0. Select all vertices, and press Assign. Now, select both upper and lower corner vertices as shown. For these, set Weight to 1.0, and press Assign again. This will make these vertices stay where they are during the softbody simulation. In Weight Paint Mode you should see something like the image.

Next, exit EditMode and go to the Softbody panel in the Object Buttons F7. Click Enable Soft Body. Increase Grav to 9.8. Activate the Use Goal button. Click the selector button next to Use Goal and choose the name of the *Vertex Group* to use for the goal, in this case, the only choice should be the default name *Group*. Now set the edge stiffnes E Stiff to 0.9, set Mass to 0.5, Friction to 0.14 and Speed to 2. Set Aero to 40. Save your work.

- Now, you can press ALT−A to see the flag react to gravity and air resistance as it comes to rest.



Example Wind setup.

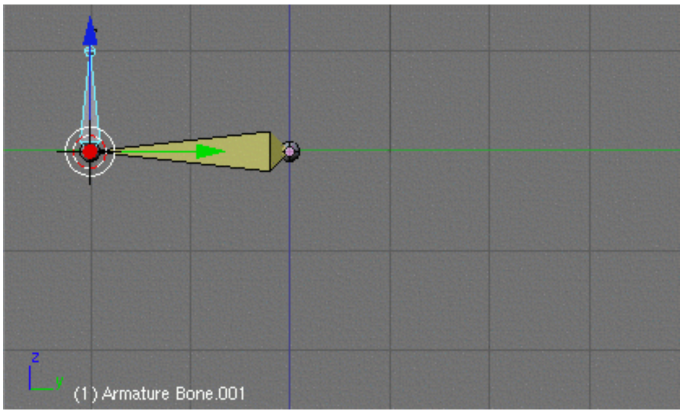


Example Wind Strength IPO.

Now we will add some wind to the simulation:

- Add an empty to the scene where the source of the wind will be. Select the Particle Interaction tab and activate the Wind button. Set Strength to 1.
- Rotate and move the empty so that the Z axis point towards the flag. See *Example Wind setup.*. You can temporarily increase the *Strength* value so that you can more easily see the effect of the wind.
- You can press ALT−A to see the flag react to wind.
- Add an IPO curve to simulate changing wind strength will add more realism to the animation. See *Example Wind Strength IPO.*.

Swing



The soft body solver is a simulation system that knows about mass and gravity. Since you can pin certain parts of the mesh to stay at a goal point, you can make a swing very easily. Add a plane and delete one half, so that all that remains are two vertices connected by an edge. Select one vertex and make it the sole member of a vertex group. Tab out of edit mode and enable soft−body for the object. Use Goal, and select the vertex group. Set goal weight to 0, and save your blend file. This creates a rig where one vertex is pinned, but the edge and the other vertex is fully affected by the simulation.

When you scrub or play (Alt A your animation, the goaled vertex stays put, but the other is free to be affected by physics. Since the edge is pretty stiff, and there is gravity and mass and thus inertia, the other vertex swings at the end, back and forth, eventually coming to a stop.

You can now parent an empty to this vertex, and the empty will swing back and forth smoothly (much better than hand animating)! Now that you have a vertex moving nicely, you can add an armature, and set the IK target of the bone to the Empty. Now the bone swings back and forth!

Jello

This example illustrates the use of a volumetric solid. A volumetric solid is one with interior edges. Ths softbody simulation works much better when there are edges inside the mesh, to give it elasticity and resist compression and collapsing of its walls.

Walking Jello

This example illustrates the use of an Armature to control the softbody.

Crash Test

Softbodies can remember their deformation by setting Plas > 0 with tensile strength simulated by setting Be > 0.

Soft Camera

A [mini−tutorial on Blender Artists] on how to mount a camera to a soft body. The resulting soft bounce and dampening effect of the soft−body simulation/movement gives a realistic sway and movement as if held by a human, where muscles do not give perfectly smooth movement.

Hints

When you enable the Soft Body effect on an object, it will always be simulated forward in time. Moving backwards in time or jumping in steps larger than 9 frames will reset the soft body back to its original position. Use the Timeline window playback to make tweaking Soft Body effects interactive.

Why does it do nothing?
: When you Enable Soft Body all of it's vertices are free floating particles in the void. This means unless any kind of interaction with 'the rest of the world' is established, they keep in the state of motion they are in (Newton's 1st law of motion). Since nothing is acting on them, they stick where they are. Reasons might be:

1. the softbody is in Edit mode,
2. you have not saved your file and thus Blender cannot save the cache
3. nothing deflects it, either other objects or forces
4. no Gravity and/or mass
5. no Edges as springs at all
6. you have Goal enabled, and the goal is to stay in the same place (no animation)

I only wanted a little jiggle...
: Use Goal. It connects the free particles with their siblings (vertices in a mesh, but curves and lattices should work too) by establishing a 'damped spring'. Use G Min and G Max to get it as close as you want.

But I want it to fall down
: Gravity or animation makes things fall down. Do not use Goal if there is no animation.

I want some vertices to be soft...while others stay
: Then you need to to have a vertex group assigned to goal.
Weight painting in that group means 1.0 sticks extacly to what comes out of Modifier stack for

'sibling' 0.0 move free in terms of .. i'll care for any force but Goal. Using weight painting, you can restrict the freedom of the soft−body mesh to act like a soft body, making it possible to "adhere" to the parented mesh, for example the "scalp−ends" of the hair stick to the head while letting the "strands" move freely. Weight−painting is the "glue" that holds the wig in place. In the pic, red is full weight (1.0) to hold the "cap" portion in place and transitions to dark blue (0.0) along the length of the hair strand strips. It's OK to use the extremes of weight here −− the soft−body settings allow you to tweak the min/max of the weighting without further painting.

My cloth goes right through other 'hard' objects
: You have to define those other objects as obstacles, and they have to exist on a common layer with the cloth. Increase your MinStep setting (MinsS on the Solver panel).

My deflection object cuts through pieces of my cloth
: You also have to set tolerances (distance and timing) to tell Blender how much to compute to avoid that intersection. See Soft Body Collision. Increase your MinStep setting (MinsS on the Solver panel).

There's like a gap of air between my Softbody and the collision object
: Reduce the Inner/Outer settings on the collision object. If that results in corners and edges cutting through your cloth again, enable CEdge. If the issue continues, enable CFace. consider making seams and edges that give the body some thickness. Change camera angle.

My cloth crumples up like paper
: Try using the Old method. Increase dampening, choke and ball size.

My cloth is jumpy and jittery
: There may be two issues: the edge springs are not being dampened (E Damp), or when a Collision is detected, Blender tries to resolve the collision too quickly. In this case you need to increase the MinS so that more checks are done, reduce the Error Limit during Collisions so that the collision is detected and prevented, or decrease the Fuzzy or increase the Choke on the response.

My cloth slides off the table
: Increase Friction.

My cloth bounces off the table
: Increase Dampening on the table (deflection object)

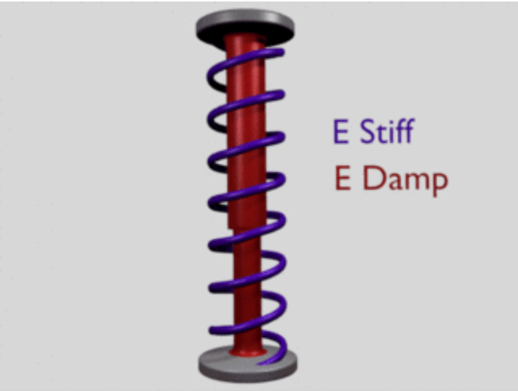
Gosh this takes a long time to compute!
: Reduce the number of vertices in your Softbody mesh. Disable CFace.

The density of a mesh increases compute time. A mesh that is not dense enough looks blocky (use Subsurf).

I made changes, but it doesn't seem to be doing anything
: Clear the cache. Ensure that Blender can create the cache files, and that the files are not write−protected.

My ball comes to a rest, but then it just jitters, like Brownian motion
: Increase the MinS setting so that it stops over−correcting vertex location.

Technical Details



Shock absorber description

In the softbody world vertices of meshes, lattices, curves .. are treated as particles having a mass. Their movement is determined by the forces affecting them. Beside other forces the individual particles can interact with another along edges using a physical model which is very close to shock absorbers used in cars. The working parts are:

- A spring trying to keep the particles at a certain distance. How hard the spring tries to do that is controlled by the softbody parameter 'E Stiff'.
- A damping element to calm the movement down. The resistance the element builds up against motion is controlled by the softbody parameter 'E Damp'.

Softbody goal

- There is another 'shock absorber' at each vertex of the softbody connecting the associated particle with the 'original' position of the vertex. So this defines a 'goal' the particle tries to reach. The strength of the springs here however is modified by either object global settings in the softbody panels or weight painted to a vertex group.
- A very special goal relation is obtained when the weight of the vertex is exactly 1. In this case the particle is "bolted" to the original vertex. The motion of that particle is the same as if it was no softbody at all. Defining goal weights smaller than 1 will cause the softbody to jiggle around it's "rest position".
- Having said that, it's not very surprising that a weight of 0 breaks the goal 'shock absorber' completely.

Clever solver

To get things done as fast as possible the solver used here follows a strategy called adaptive step sizing and works like this:

- First try to warp to frame N+1 (or shorter if we were told (MinS will do so)) with all the knowledge we have.
- Use some voodoo to detect how bad we were doing.
- If we did not violate the level of badness defined by the error limit we 're done and will happily return our results to the boss.

- Other ways we failed to hold the badness line go to plan B.
- Plan B:

- Assume there happened something we need to look at closer.
- Reduce the warp distance, if we are allowed to do so. (MaxS or internal emergency break will tell)
- Try to warp to reduced distance with all the knowledge we have.
- Use some voodoo to detect how bad we were doing.
- If we did not violate the level of badness defined by the error limit we continue trying to warp to next step with reduced distance with all the knowledge we have until we reach destination. Once hopefully there return to boss with a smile.
- Here we start to get optimistic: if voodoo tells us we were doing pretty good we'd even could try to go faster then!
- If we could not meet the level of badness still we will follow plan B recursively provided we are allowed to do so(MaxS or internal emergency break will tell)
- When we were not allowed to decrease step size any more (MaxS or internal emergency break did say), we will do our very best and return the poor results to the boss.

Note: you can kick, fire, kill your computer it won't give better results.

Baking Soft Bodies

V.: pre−2.50

Mode: Object Mode

Panel: Physics Context Soft Body

Description

Pre−2.50, you had to manually bake the simulation. This section is preserved for those people using an old version of Blender.

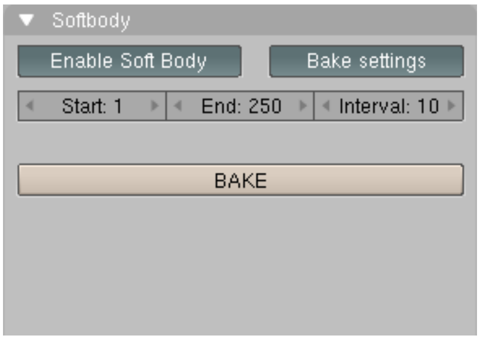
Once you've got a working soft body simulation, you can Bake the simulation into a static animation system. A baked soft body plays back much more quickly, and is not dependent on before and after frames any more.

It is recommended that you bake soft bodies when rendering animations, because the simulation doesn't work correctly for Motion Blur rendering, or for rendering in small chunks via a network render system.

Cannot Bake in Edit Mode

You can only bake when in object mode. Clicking Bake with the object in edit mode won't appear to do something, but won't actually calculate anything.

Options



Bake settings.

Start/End
: Sets the range of the soft body simulation to be baked.

Interval
: Sets the number of frames in between each baking "step" (the "resolution" of the baked result). Intermediate positions will be calculated using the steps as key frames, with B Spline interpolation.

Bake
: Starts the Bake process. Depending on complexity, this might take a little while. You can press Esc to stop baking. Once Baked, this button changes into a "Free Bake" button. You must free the baked result to modify soft body settings.

Developer Notes

Previous: Manual/Static Particles	Contents	*Next:* Manual/Force Fields and Deflection

User Manual: Contents | Guidelines | Blender Version **2.34**

Force Fields

Mode: Object Mode

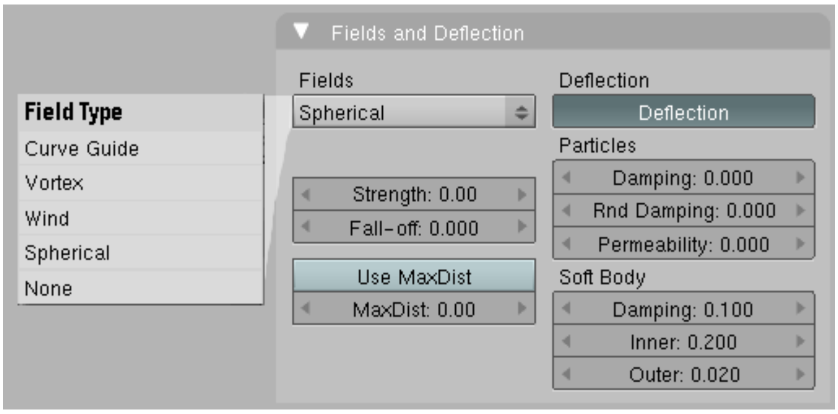
Panel: Object/Physics Context Fields & Deflection

Hotkey: F7

Description

Blender's physics system has a set of force fields that can influence a dynamic simulation (particles, hair, or soft bodies). Force fields will act upon points within a certain area, and influence their movement in various ways. Any object can be used as a force field, though Empties are the most common since they aren't rendered.

Options

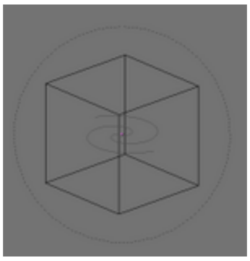


Fields and Deflection panel.

All force fields have the same options (except Curve Guides, see the chapter on the static particles):

Strength
: The strength of the field effect. This can be positive or negative to change the direction that the force operates in. A force field's strength is scaled with the force object's scale, allowing you to scale up and down scene, keeping the same effects.

Fall−off
: How much the strength diminishes with distance from the force field object



Max distance indicator

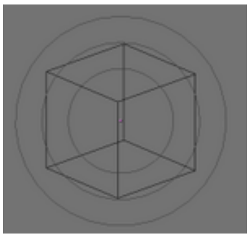
Use MaxDist
: Makes the force field only take effect within a specified maximum radius (shown by a dashed circle around the object)

MaxDist
: The radius distance to affect points within

For all options for force fields, except MaxDist parameter, Ipo keys can be inserted. The Ipo curves (FStreng and FFall) are edited as Object Ipo types in the Ipo window. See the chapter on the [Animation Basics] for more on Animation and IPO.

Force fields come in a few different types:

Spherical

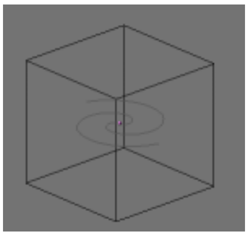


Spherical field indicator

Gives a constant force towards (positive strength) or away from (negative strength) the object's center. This acts like gravity, sucking points towards the object, or repelling them away.

Spherical force fields are much faster to calculate than mesh−based deflection, and in cases where accuracy isn't of prime importance, they can be used with MaxDist and a negative strength for a much faster alternative for collisions.

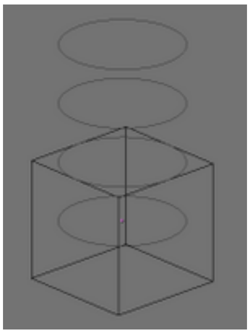
Vortex



Vortex field indicator

Vortex fields give a spiralling force that twists the direction of points around the force object's local Z axis. This can be useful for making a swirling sink, or tornado, or kinks in particle hair.

Wind



Wind field indicator

Wind gives a constant force in a single direction, along the force object's local Z axis. The strength of the force is visualised by the spacing of the circles shown

Hints

In trying to debug why the wind isn't blowing, check:

- Is the Wind Empty on the same Layer as the Particle Emitter?
- Do you have MaxDist on for the Wind but not a high enough distance value?
- Are your Wind circles pointing in the right direction?

Deflection

Mode: Object Mode

Panel: Object/Physics Context Fields & Deflection

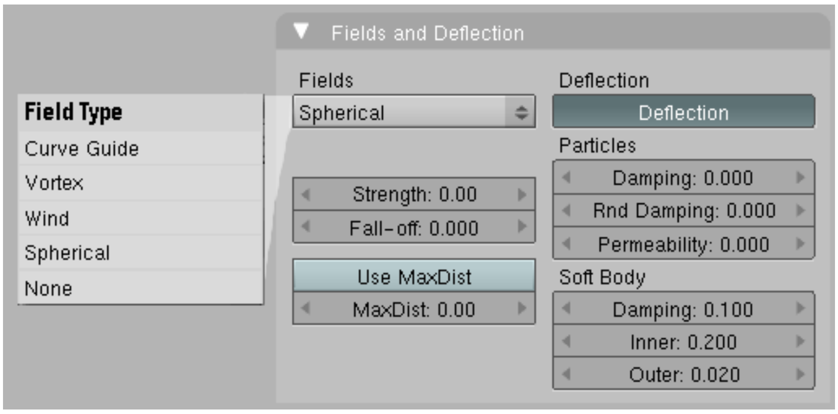
Hotkey: F7

Description

As well as force objects, any mesh object can be set as a deflector. Particles will then bounce on the surface of the mesh. You can control how much the particles bounce with the Damping value, add some randomness to the bounce with Rnd Damping, and you can define the percentage of particles which pass through the mesh with the Permeability parameter.

You will not see any extra graphic indicators with deflectors as you do with force fields.

Options



Fields and Deflection panel.

Deflectors have two sets of options, for particle deflection and softbody deflection:

Particles

Damping
: The amount of bounce that surfaces will have, ranging from:
  - ◊ 0.0 − No damping, particles will have maximum bounce, to
  - ◊ 1.0 − Maximum damping, particles will not bounce at all

Rnd Damping
: Adds a random variability to the bounce. For example, with a Damping of 1.0 and a Rnd Damping of 0.5, the damping will vary between 1.0 and 1.5.

Permeability
: Percentage of particles passing through the mesh, ranging from:
  - ◊ 0.0 − No particles pass through, to
  - ◊ 1.0 − All particles pass through the deflector

Soft Body

Damping
: The amount of bounce that surfaces will have, ranging from:
  - ◊ 0.0 − No damping, soft bodies will have maximum bounce, to
  - ◊ 1.0 − Maximum damping, soft bodies will not bounce at all.

Inner / Outer

An artificial padding distance added to the inside and outside of each face, to help prevent intersections

If you set up a particle deflector you'll have to make sure sufficient keys are available for Blender to calculate the collisions with sufficent detail. If you see particles moving through your deflector or bouncing in the wrong positions, then there might be problem with too few keys (Keys setting, Particle Motion tab) or your particle/deflector is moving too fast.

For the options from the Particles group, Ipo keys can be inserted. The Ipo curves (Damping, RDamp and Perm) are edited as Object Ipo types in the Ipo window. See the chapter on the Animation Basics for more on Animation and IPO.

Examples

Here is a Meta object, dupliverted to a particle system emitting downwards, and deflected by a mesh cube:

Modified end result.

Hints

- Make sure that the normals of the mesh surface are facing towards the particles/points for correct deflection.
- You can animate moving deflectors but particles can leak through the mesh if the deflector moves to fast or if the mesh is complicated. This can be partly solved by increasing the Keys parameter for the particle emitter.
- After changing any parameters, you will need to select your particle emitter and go back to the *Particles* tab and press *RecalcAll* button.
- More keys means longer calculation times and usage of more memory. See the section called Particles for how to set up particle emitters.

Limitations & Work−Arounds

Currently (in Blender 2.42) static particles ignore mesh deflectors.

Previous: Manual/Soft Bodies | Contents | *Next:* Manual/Rigid Bodies

User Manual: Contents | Guidelines | Blender Version **2.41**

Starting with 2.41 you can bake game engine hard body dynamics into animation IPO curves.

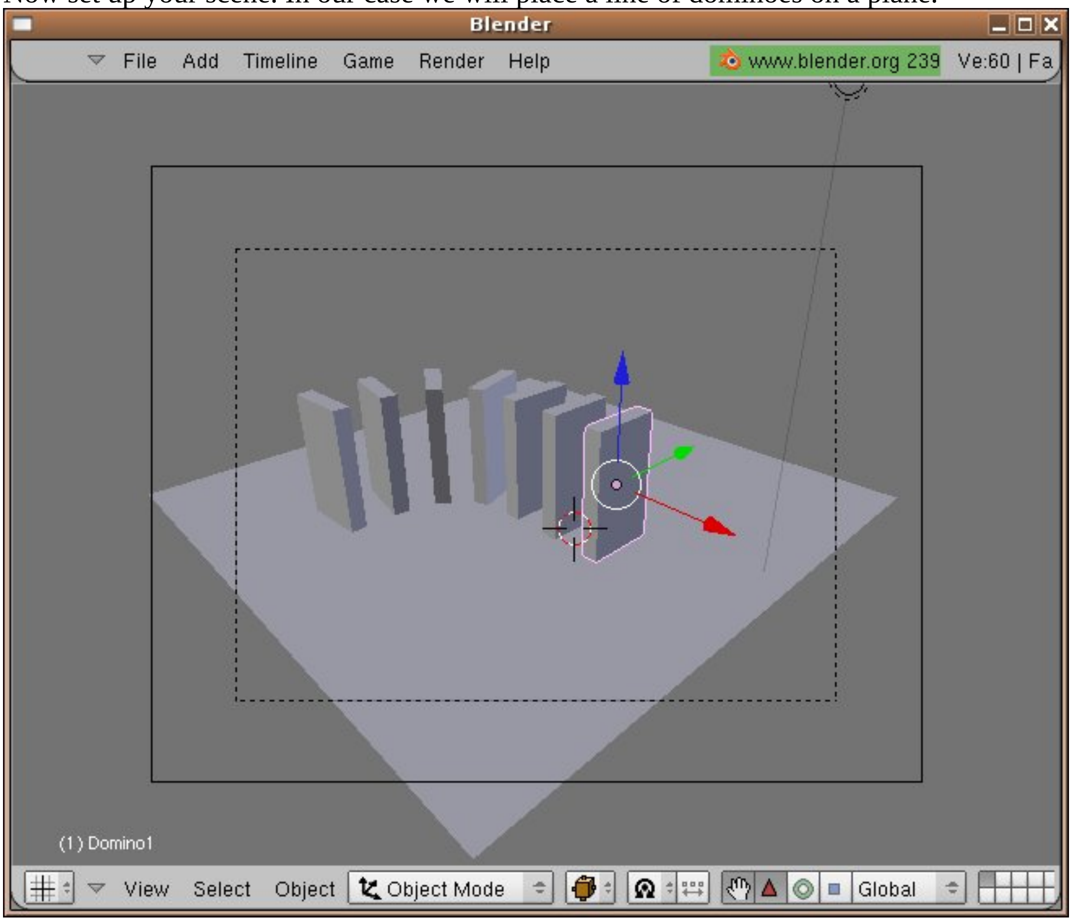
setting up world physics

First set up the world physics.

Go to Material context (F5) [1] then the World buttons [2], and control that the physics engine is set to Bullet [3]:

setting up your scene

Now set up your scene. In our case we will place a line of dominoes on a plane.



Create a plane (Add−>Mesh−>Plane) and some dominoes (Add−>Mesh−>Cube) arranged as shown.

setting up collision objects

Next select one of the objects that will collide − in our case the first of the dominoes in the line. Tilt it toward the second (rotate) − we'll start it tipping over and let Blender do the rest.

Now for each of the objects (in this case, the dominoes) that will collide

Go to the logic button and click on Actor

Click Dynamic

Click RigidBody

Click Bounds

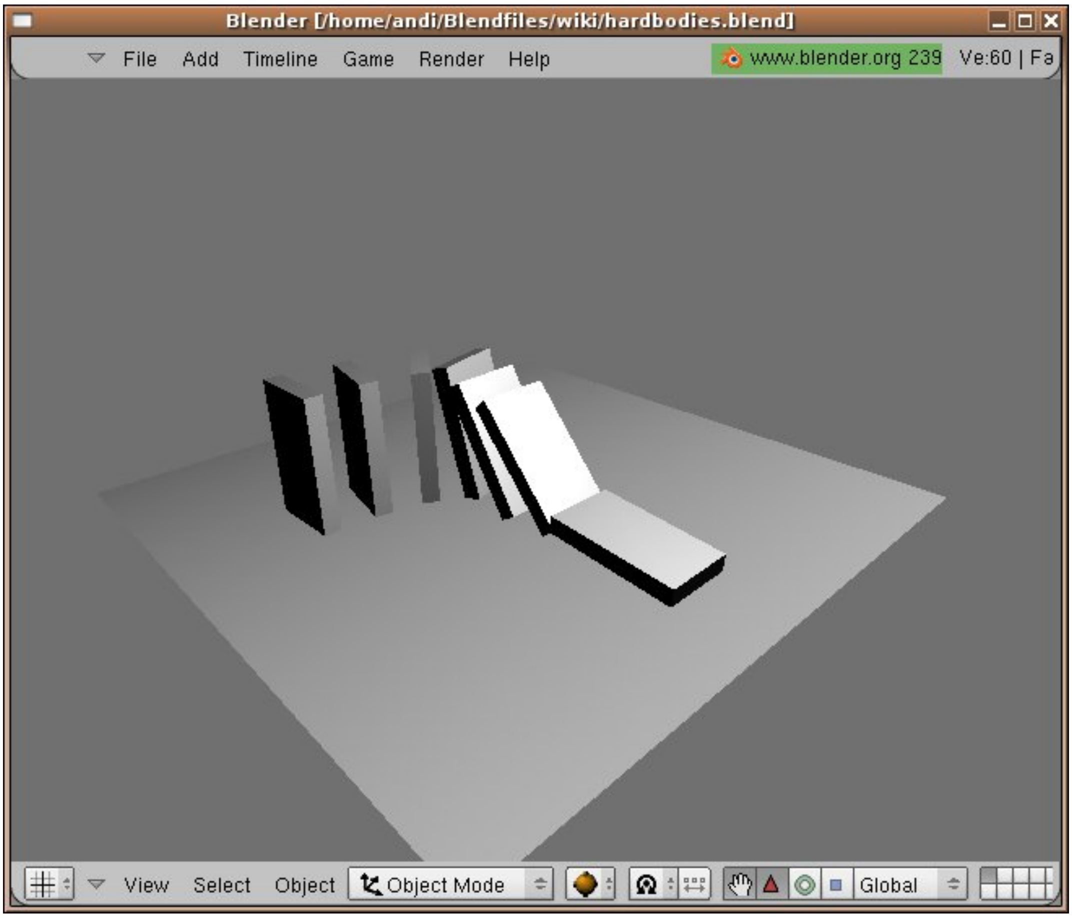
Change the Bounds type to Convex Hull Polytope (other types can be done but are more complex to setup)

Do this for each object that will be involved in the collisions. The plane (ground) in this case should not have its Logic actor & bounds set. Although it does interact with the other objects (in this case, supporting the dominoes), if we set it as a dynamic it too would fall away by gravity. So, it will be a static actor.

Also do not forget that physics objects **have** to have a material, or they will bounce around wildly because they need the damping values etc. which are not set on objects without materials.

animation preview

Press P to see a preview of the game animation

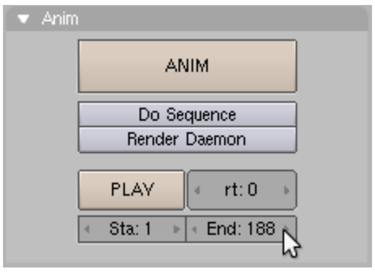


Press ESC when done

get ready to bake

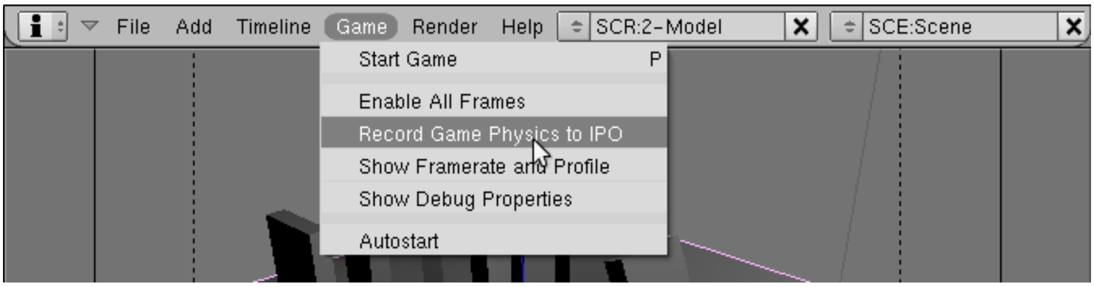
Go to the Scene buttons ( or F10).

Select the Keyframe start and end range you want the animation you record to start and end on.



record the IPO

Select Game Record Game Physics to IPO:

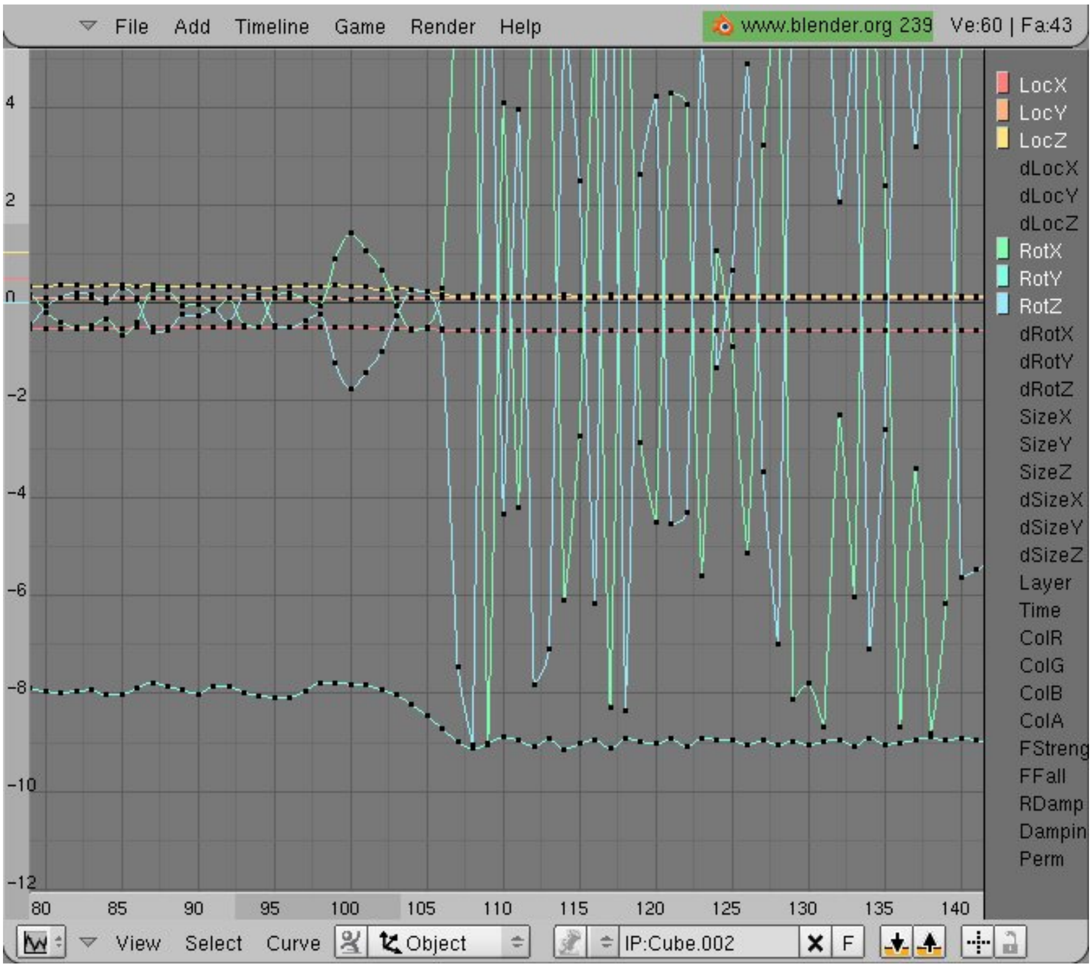


press the P

press ESC when done

scrub through the animation

now you can scrub forward and backwards through your animation with the arrow keys, and look at the animation curves in ipo window by selecting an object.



replace the animation

If you change something and press the P the old animation will be recorded over if you have record IPO still enabled.

troubleshooting and hints

As with all physics systems there are a number of limitations that you need to be aware of please see this page for tips, tricks, and limitations
http://www.continuousphysics.com/mediawiki−1.5.8/index.php?title=Physics_Tips

Previous: Manual/Force Fields and Deflection	Contents	*Next:* Manual/Fluid Simulation

User Manual: Contents | Guidelines | Blender Version **2.45**

Fluid Simulation

Mode: Object Mode / Edit Mode (Mesh)

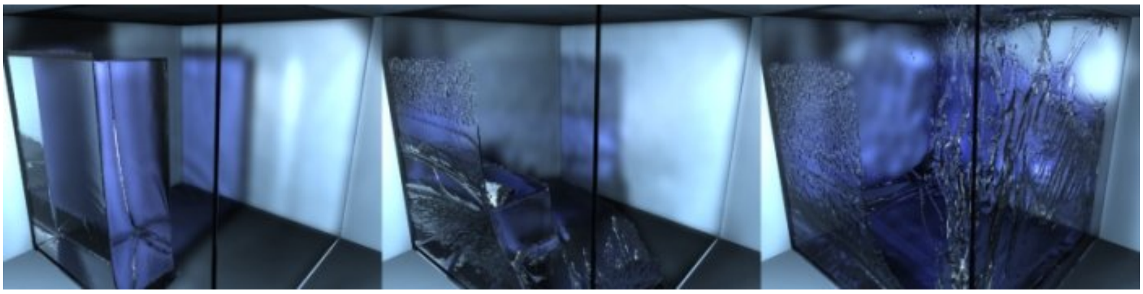
Panel: Physics Context Fluid Simulation

Hotkey: F7

Description

While modeling a scene with blender, certain objects can be marked to participate in the fluid simulation, e.g. as fluid or as an obstacle. The bounding box of another object will be used to define a box−shaped region to simulate the fluid in (the so called simulation domain). The global simulation parameters (such as viscosity and gravity) can be set for this domain object.

Using the "bake" button, the geometry & settings are exported to the simulator and the fluid simulation is performed, generating a surface mesh together with a preview for each animation frame, and saving them to hard disk. Then the appropriate fluid surface for the current frame is loaded from disk and displayed or rendered.



A breaking dam example

Process

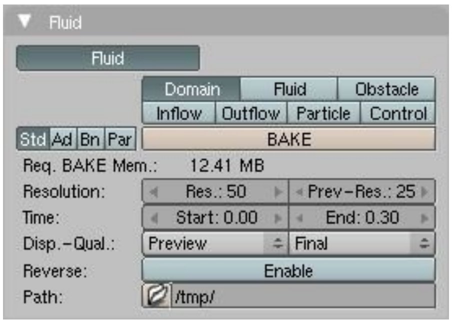
In general, you follow these steps:

◊ 1. Model the scene (objects, materials, lights, camera).
◊ 2. Designate the portion of the scene where the fluid will flow (the domain).
◊ 3. Specify the functions of the various objects as they relate to the fluid (an inlet, outlet, or obstruction, etc.).
◊ 4. Create the fluid source(s), and specify its (their) material, viscosity, and initial velocity.
◊ 5. Bake in a preliminary simulation.
◊ 6. Revise as necessary, saving changes.
◊ 7. Bake in a final simulation.

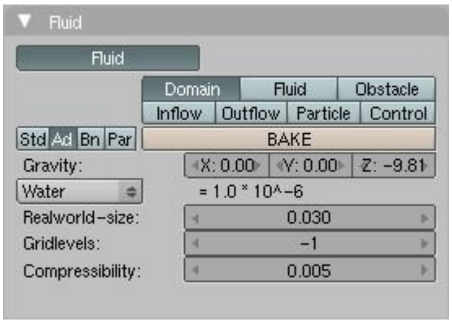
When you Bake, the version identifier in the User Preferences header changes to a status bar, showing the number of frames baked and the total number to go. Baking takes a LOT of compute power (and time). Don't watch it, because we all know that a watched pot does not boil, a watched cake does not bake, and watching fluid bake is like watching grass grow. Baking is best done overnight.

Because of the possibility of spanning and linking between scenes, there can only be one domain in an entire .blend file.

Options



The basic (and frequently needed) fluid simulation options



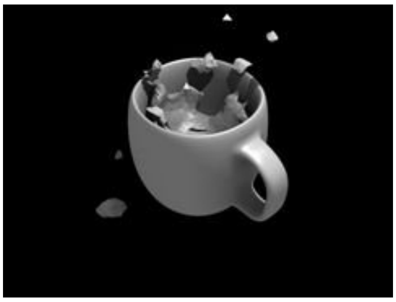
Less frequently needed advanced options

Domain/Fluid/Obstacle/Inflow/Outflow
: Selecting one of these buttons determines how the enabled object will be used during the simulation. Each button determines a different functionality/interaction, and each has different further options that will become available.

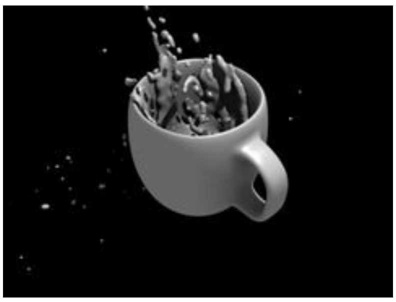
Mesh rendering

If the mesh has modifiers, the rendering settings are used for exporting the mesh to the fluid solver. Depending on the setting, calculation times and memory use might exponentially increase. For example, when using a moving mesh with subsurf as an obstacle, it might help to decrease simulation time by switching it off, or to a low subdivision level. When the setup/rig is correct, you can always increase settings to yield a more realistic result.

Domain



10cm mug at Resolution 70



10cm mug at Resolution 200

The bounding box of the object serves as the boundary of the simulation. **All fluid objects must be in the domain.** Fluid objects outside the domain will not bake. No tiny droplets can move outside this domain; it's as if the fluid is contained within the 3−D space by invisible force fields. There can be only a single fluid simulation domain object in the scene. The lengths of the bounding box sides can be different.

Domain Space

The shape of the object does not matter because it will always be a fixed cubic of space, so usually there won't be a reason to use any other shape than a box. If you need obstacles or other boundaries than a box to interfere with the fluid flow, you need to insert additional obstacle objects inside the domain boundary.

Resolution
: The granularity at which the actual fluid simulation is performed. This is probably the most important setting for the simulation as it determines the amount of detail in the fluid, the memory and disk usage as well as computational time. Note that the amount of required memory quickly increases: a resolution of 32 requires ca. 4MB, 64 requires ca. 30MB, while 128 already needs more than 230MB. Make sure to set the resolution low enough, depending on how much memory you have, to prevent Blender from crashing or freezing.

  Be sure to set the resolution appropriate to the real−world size of the domain (the actual physical size of the domain is set in the advanced settings). If the domain is not cubic, the resolution will be taken for the longest side. The resolutions along the other sides will be reduced according to their lengths (note that a non−cubic domain will therefore need less memory than a cubic one, resolutions being the same).

Preview−Res
: This is the resolution at which the preview surface meshes will be generated. So it does not influence the actual simulation, and even if there is nothing to see in the preview, there might be a thin fluid surface that cannot be resolved in the preview.

Start time
: Simulation start time (in seconds). This option makes the simulation computation in Blender start later in the simulation. The domain deformations and fluid flow prior to that is not saved.

  For example, if you wanted the fluid to appear to already have been flowing for 4 seconds before the actual first frame of data, you would enter 4.0 here.

End time
: Simulation ending time (in seconds). If your frame−rate is 25 frames per second, and ending time is 4.0 seconds, then you should (if your start time is 0) set your Animation to end at frame 100. Baking always respects the end frame set in the ANIM panel. Start and end times have nothing to do with how many frames are baked, but instead are based on physical force and fluid viscosity.

  If you set Start Time to 3.0, and End Time to 4.0, you will simulate 1 second of fluid motion. That one second of fluid motion will be spread across however many frames are set in the Animation panel in your End: setting. The fluid simulator disregards the Sta: setting in the ANIM panel, it will always bake from frame 1.

Starting a Simulation on Frames Other Than 1

If you wish the simulation to start later than frame 1, you must key the fluid objects in your domain to be inactive until the frame you desire to start the simulation. See section 1.4 'Animating Fluid Property Changes' for more information.

> If you have Blender set to make 250 frames at 25 fps (Scene buttons, Render context, Anim panel and Output Panel), the fluid will look like it had already been flowing for 3 seconds at the start of the simulation, but will play in slow motion, since the 1 second fluid sim plays out over the course of 10 video seconds. If you change the end time to 13 to match the 250 frames at 25 fps, the simulation will be real−time since you set 10 seconds of fluid motion to simulate over 10 seconds of animation. Having these controls in effect gives you a "speed control" over the simulation.

Disp.−Qual
: How to display a baked simulation in the Blender GUI (first pulldown menu) and for rendering (second one): original geometry, preview mesh or final mesh. When no baked data is found, the original mesh will be displayed by default.

Displaying a Baked Domain

After you have baked a domain, it is displayed (usually) in the Blender window as the preliminary mesh. To see the size and scope of the original domain box, select Geometry in the left dropdown.

Bake directory
: **REQUIRED** Directory and file prefix to store baked surface meshes with. This is similar to the animation output settings, only selecting a file is a bit special: when you select any of the previously generated surface meshes (e.g. `untitled_OBcube_fluidsurface_final_0132.bobj.gz`), the prefix will be automatically set (`untitled_OBcube_` for this example). This way the simulation can be done several times with different settings, and allows quick changes between the different sets of surface data.

BAKE
: Perform the actual fluid simulation. The blender GUI will freeze and only display the current frame that is simulated. Pressing Esc will abort the simulation. Afterwards two `.bobj.gz` (one for the final quality, one for the preview quality), plus one `.bvel.gz` (for the final quality) will be in the selected output directory for each frame.

  Note on freeing the previous baked solutions: Deleting the content of the Bake directory is a destructive way to achieve this, be careful if more than one simulation uses the same bake directory (be sure they use different filenames, or they will overwrite one another).

Reusing Bakes

Manually entering (or searching for) a previously saved (baked) computational directory and filename mask will switch the fluid flow and mesh deformation to use that which existed during the old bake. Thus, you can re−use baked flows by simply pointing to them in this field.

Selecting a Baked Domain

After a domain has been baked, it changes to the fluid mesh. To re−select the domain so that you can bake it again after you have made changes, go to any frame and select (right−click) the fluid mesh. Then you can click the Bake button again to recompute the fluid flows inside that domain.

St/Ad/Bn/Par−Button

Clicking this button will show other panels (Standard/Advanced/Boundary) of more advanced options, that often are fine set at the defaults.

Advanced

Gravity vector

Strength and direction of the gravity acceleration and any lateral (x,y plane) force. The main component should be along the negative z−axis [m/s^2]. All of the x,y,z values should not be zero, or the fluid won't flow (imagine a droplet in space). It must be some small number in at least one direction.

Viscosity

The "thickness" of the fluid and actually the force needed to move an object of a certain surface area through it at a certain speed. You can either enter a value directly or use one of the presets in the drop down (such as honey, oil, or water). For manual entry, please note that the normal real−world viscosity (the so−called dynamic viscosity) is measured in Poiseuille (pronounced *"pwazooze"*) units, and commonly centiPoise ('"sentipwaz'") units (cP). Blender, on the other hand, uses the kinematic viscosity (which is dynamic viscosity divided by the density, unit [m^2/s]). The table below gives some examples of fluids together with their dynamic and kinematic viscosities.
Thus, manual entries are specified by a floating point number and an exponent. These floating point and exponent entry fields (scientific notation) simplify entering very small or large numbers. The viscosity of water at room temperature is 1.002 cP; so the entry would be 1.002 times 10 to the minus six (10^−6). Hot Glass and melting iron is a fluid, but very thick; you should enter something like 1 x 10^0 as its viscosity (indicating a value of 1x10^6 cP). Note that the simulator is not suitable for non−fluids, such as materials that do not "flow". Simply setting the viscosity to very large values will not result in rigid body behavior, but might cause instabilities.

Viscosity Varies

The default values in Blender are considered typical for those types of fluids and "look right" when animated. However, actual viscosity of some fluids, esp. sugar−laden fluids like chocolate syrup and honey, depend highly on temperature and concentration. Oil viscosity varies by SAE rating. Glass at room temperature is basically a solid, but glass at 1500 degrees flows like water.

Blender Viscosity Unit Conversion

Fluid	dynamic viscosity	kinematic viscosity (Blender)
Water (20Â°C)	1.002 (1x10^0)	1 x 10^−6 (.000001)
Oil SAE 50	500 (5x10^2)	5 x 10^−5 (.00005)
Honey (20Â°C)	10,000 (1x10^4)	2 x 10^−3 (.002)
Chocolate Syrup	30,000 (3x10^4)	3 x 10^−3
Ketchup	100,000(1x10^5)	1 x 10^−1
Melting Glass	1x10^15	1 x 10^0

Real−World size

Size of the domain object in the real world in meters. If you want to create a mug of coffee, this might be 10 cm (0.1 meters), while a swimming pool might be 10m. The size set here is for the longest side of the domain bounding box.

Gridlevel

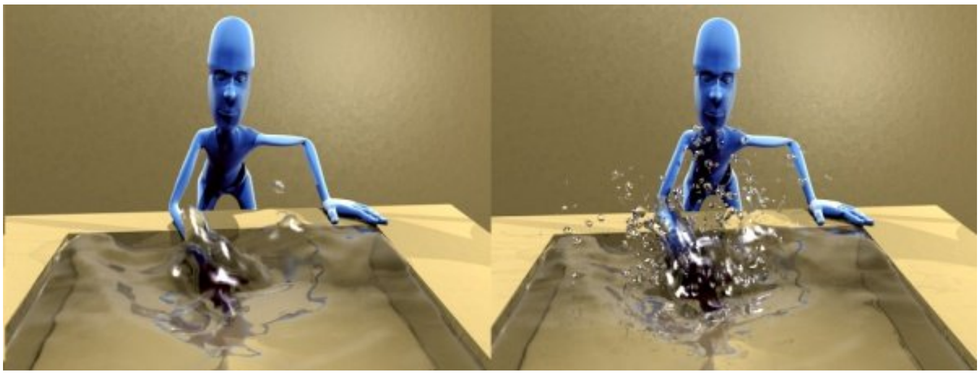
How many adaptive grid levels to be used during simulation − setting this to −1 will perform automatic selection.

Compressibility
: If you have problems with large standing fluid regions at high resolution, it might help to reduce this number (note that this will increase computation times).

Domain boundary type settings
: This is the same as for obstacle objects below, it will basically set the six sides of the domain to be either sticky, non−sticky, or somewhere in between (this is set by the PartSlipValue).

Tracer Particles
: Number of tracer particles to be put into the fluid at the beginning of the simulation. To display them create another object with the `Particle` fluid type, explained below, that uses the same bake directory as the domain.

Generate Particles
: Controls the amount of fluid particles to create (0=off, 1=normal, >1=more). To use it, you have to have a surface subdivision value of at least 2.



An example of the effect of particles can be seen here − the image to the left was simulated without, and the right one with particles and subdivision enabled.

Surface Smoothing
: Amount of smoothing to be applied to the fluid surface. 1.0 is standard, 0 is off, while larger values increase the amount of smoothing.

Surface Subdivision
: Allows the creation of high−res surface meshes directly during the simulation (as opposed to doing it afterwards like a subdivision modifier). A value of 1 means no subdivision, and each increase results in one further subdivision of each fluid voxel. The resulting meshes thus quickly become large, and can require large amounts of disk space. Be careful in combination with large smoothing values − this can lead to long computation times due to the surface mesh generation.

Generate & Use SpeedVecs
: If this button is clicked, no speed vectors will be exported. So by default, speed vectors are generated and stored on disk. They can be used to compute image based motion blur with the compositing nodes.

Fluid

All regions of this object that are inside the domain bounding box will be used as actual fluid in the simulation. If you place more than one fluid object inside the domain, they should currently not intersect. Also make sure the surface normals are pointing outwards. In contrast to domain objects, the actual mesh geometry

is used for fluid objects.

Volume Init Type
: `Volume Init` will initialize the inner part of the object as fluid. This works only for closed objects. `Init Shell` will initialize only a thin layer for all faces of the mesh. This also works for non closed meshes. `Init Both` combines volume and shell; the mesh should also be closed. See the picture below.

Initial velocity
: Speed of the fluid at the beginning of the simulation in meters per second.

Example of the different volume init types: volume, shell and both. Note that the shell is usually slightly larger than the inner volume.

Obstacle

This object will be used as an obstacle in the simulation. As with a fluid object, obstacle objects currently should not intersect. As for fluid objects, the actual mesh geometry is used for obstacles. For objects with a volume, make sure that the normals of the obstacle are calculated correctly, and radiating properly (use the Flip Normal button, Mesh Tools panel, Editing context [F9]), particularly when using a spinned container. Applying the Modifier Subsurf before baking the simulation could also be a good idea if the mesh is not animated.

Volume Init Type
: Same as for a fluid object above.

Boundary Type (see picture below)
: Determines the stickiness of the obstacle surface, called Surface Adhesion. Surface Adhesion depends in real−world on the fluid and the graininess or friction/adhesion/absorbsion qualities of the surface.
  - ◊ `Noslip` causes the fluid to stick to the obstacle (zero velocity),
  - ◊ `Free(−slip)` allows movement along the obstacle (only zero normal velocity),
  - ◊ `Part(−slip)` mixes both types, with 0 being mostly noslip, and 1 being identical to Freeslip. Note that if the mesh is moving, it will be treated as noslip automatically.

Animated Mesh
: Click this button if the mesh is animated (e.g. deformed by an armature, shape keys or a lattice). Note that this can be significantly slower, and is not required if the mesh is animated with position or rotation IPOs.

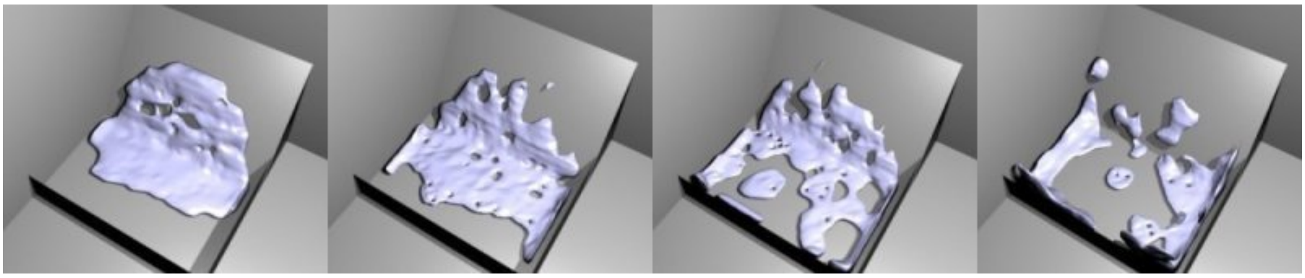
PartSlip Amount
: Amount of mixing between no− and free−slip above.

Animated Objects are No Slip:

If an obstacle is moving, Blender treats it automatically as no slip (sticky). If you want fluid to splash off of a moving object, put an transparent plane in the spot where the fluid will hit the

moving object, exactly aligned and shaped as the object, but just a teeny bit in front of the object between the object and the fluid. As the object whizzes by and the fluid splashes, the splash will actually contact the stationary transparent plane and slide off, and the object will continue on its merry way.

Impact Factor
: Amount of fluid volume correction for gain/loss from impacting with moving objects. If this object is not moving, this setting has no effect. However, it if is and the fluid collides with it, a negative value takes volume away from the Domain, and a positive number adds to it. Safe ranges or −3 to 3.0.



Example of the different boundary types for a drop falling onto the slanted wall. From left to right: no−slip, part−slip 0.3, part−slip 0.7 and free−slip.

Inflow

This object will put fluid into the simulation (think of a water tap).

Volume Init Type
: Same as for a fluid object above.

Inflow velocity
: Speed of the fluid that is created inside of the object.

Local Inflow Coords
: Use local coordinates for the inflow. This is useful if the inflow object is moving or rotating, as the inflow stream will follow/copy that motion. If disabled, the inflow location and direction do not change.

Outflow

Any fluid that enters the region of this object will be deleted (think of a drain or a black hole). This can be useful in combination with an inflow to prevent the whole domain from filling up.

Volume Init Type
: Same as for a fluid object above. When enabled, this is like a tornado (waterspout) or wet vac" vacuum cleaner, and the part where the fluid disappears will follow the object as it moves around.

Particle

This type can be used to display particles created during the simulation. For now only tracers swimming along with the fluid are supported. Note that the object can have any shape, position or type − once the particle button is pressed, a particle system with the fluid simulation particles will be created for it at the correct position. When moving the original object, it might be necessary to delete the particle system, disable the fluidsim particles, and enable them again. The fluidsim particles are currently also unaffected by any other particle forces or settings.

Drops
: Surface splashes of the fluid result in droplets being strewn about, like fresh water, with low Surface Tension.

Floats
: The surface tension of the fluid is higher and the fluid heavier, like cold seawater and soup. Breakaways are clumpier and fall back to the surface faster than Drops, as with high Surface Tension.

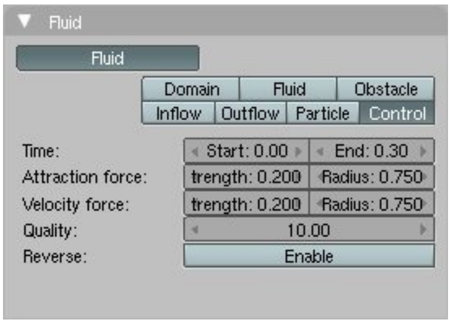
Tracer
: Droplets follow the surface of the water where it existed, like a fog suspended above previous fluid levels. Use this to see where the fluid level has been.

Size Influence
: The particles can have different sizes, if this value is 0 all are forced to be the same size.

Alpha Influence
: If this value is >0, the alpha values of the particles are changed according to their size.

Bake directory
: The simulation run from which to load the particles. This should usually have the same value as the fluid domain object (e.g. copy by ctrl−c, ctrl−v).

Control



Fluid control options

Time
: You specify the start and end time during which time the fluid control object is active

Attraction force
: The attraction force specifies the force which gets emmited by the fluid control object. Positive force results in attraction of the fluid, negative force in avoidance.

Velocity force
: If the fluid control object moves, the resulting velocity can also introduce a force to the fluid.

Quality
: Higher quality result in more control particles for the fluid control object

Reverse
: The control particle movement gets reversed

Look here for more information.

Another example animation of a falling drop, simulated in Blender and rendered in Yafray

Animating Fluid Property Changes

A new type of IPO Curve, FluidSim, is available for fluid domain objects. Unlike most other animatable values in Blender, FluidSim IPOs cannot be keyframed by simply using the I key; you must manually set values by clicking in the IPO window. In order to set a keyframe, you must select the property you wish to animate in the IPO window and Ctrl LMB click to set the keyframe to the desired location in the IPO window.

Enter Properties:

Note that you do not have to be exact on where you click; we recommend that after you set the control point, open the Properties (N−key) window and round the X value to a whole frame number, and then set the Y value that you wish.

The fluid domain has several channels that control the fluid over time:

Fac−Visc
: A multiplicative factor in the fluid's viscosity coefficient. It must set it before baking, and changes the viscosity of the fluid over time, so you can turn water in oil.

Fac−Time
: Changes the speed of the simulation; like the Speed Control in the VSE can speed up or slow down a video, this curve can speed up or slow down the fluid motion during a frame sequence. If the value for Fac−Tim is less than or equal to zero, time (and the fluid) stands still; the fluid freezes. For values between 0.0 and 1.0, the fluid runs slower and will look thicker. 1.0 is normal fluid motion, and values greater than 1.0 will make the fluid flow faster, possibly appearing thinner.

Gravity
: The XYZ vector for gravity changes; aka inertia of the fluid itself. Think drinking a cup of coffee while driving NASCAR at Talladega, or sipping an espresso on the autobahn, or watering the plants on the Space Shuttle. Changes in this curve make the fluid slosh around due to external forces.

Velocity
: Spurts of water from the garden hose can be simulated via this curve, to mimic changes in pressure. It also can be used to simulate the effect of wind on a stream of water, for example.

Active
: When Active transitions from 0.0 to something greater than 0 (such as between 0.1 and 1.0), the object's function (designated as an Inflow, or Outflow, etc.) resumes its effect. Crossing down to 0.0 and then at some point, back up re−establishes the effect and the resulting fluid sim. Use this for dripping, or any kind of intermittent inflow. This active status also works for objects designated as Outflow and Obstacle, so you can also simulate (for example) a drain plugging up.

Technical Details

'My cup runneth over' created with Blender and Yafray

Fluid animation can take a lot of time − the better you understand how it works, the easier it will be to estimate how the results will look. The algorithm used for Blender's fluid simulation is the *Lattice Boltzmann Method* (LBM); other fluid algorithms include *Navier−Stokes* (NS) solvers and *Smoothed Particle Hydrodynamics* (SPH) methods. LBM lies somewhere between these two. In general, it is really hard for current computers to correctly simulate even a 1−meter tank of water. For simulating a wave crashing through a city, you would probably need one of the most expensive supercomputers you could get, and it might still not work properly, no matter which of the three algorithms above you're using.

Realism

Similar to what film makers have been doing in "analogue" movies for years, you can pretend to have a wave in a city by building a smaller model, have a small wave in the model at farily high resolution, and hope that nobody will notice the difference between a 100m and a 1m wave. Texture the wave front with lots of noise and clouds affecting the color. Add lots of smoke (mist) emitters on surfaces that the wave hits, timing them to emit at the moment of impact in a direction incident to the surface and collision. Animate cars and trash (and drowning people) to float and bob on the wave front using the baked mesh. Use a string of mist emitters pointing up positioned at the wave crest to simulate the mist that blows off the top of the crest into the air.

For Blender's LBM solver, the following things will make the simulation harder to compute:

- Large domains.
- Long duration.
- Low viscosities.
- High velocities.

The viscosity of water is already really low, so especially for small resolutions, the turbulence of water can not be correctly captured. If you look closely, most simulations of fluids in computer graphics do not yet look like real water as of now. Generally, don't rely on the physical settings too much (such as physical domain size or length of the animation in seconds). Rather try to get the overall motion right with a low resolution, and then increase the resolution as much as possible or desired.

Hints

- Don't be surprised, but you'll get whole bunch of mesh (.bobj.gz) files after a simulation. One set for prelim, and another for final. Each set has a .gz file for each frame of the animation. Each file contains the simulation result − so you'll need them.
- Currently these files will not be automatically deleted, so it is a good idea to e.g. create a dedicated directory to keep simulation results. Doing a fluid simulation is similar to clicking the `ANIM` button − you currently have to take care of organizing the fluid surface meshes in some directory yourself. If you want to stop using the fluid simulation, you can simply delete all the `*fluid*.bobj.gz` files.
- Before running a high resolution simulation that might take hours, check the overall timing first by doing lower resolution runs.
- Fluid objects must be completely inside the bounding box of the domain object. If not, baking may not work correctly or at all. Fluid and obstacle objects can be meshes with complex geometries. Very thin objects might not appear in the simulation, if the chosen resolution is too coarse to resolve them (increasing it might solve this problem).
- Note that fluid simulation parameters, such as inflow velocity or the active flag can be animated with fluidsim IPOs.
- Don't try to do a complicated scene all at once. Blender has a powerful compositor that you can use to combine multiple animations. For example, to produce an animation showing two separate fluid flows while keeping your domain small, render one .avi using the one flow. Then move the domain and render another .avi with the other flow using an alpha channel. Then, composite both .avi's using the compositor's add function. A third .avi is usually the smoke and mist and it is laid on top of everything as well. Add a rain sheet on top of the mist and spray and you'll have quite a storm brewing! And then lightning flashes, trash blowing around, all as separate animations, compositing the total for a truly spectacular result.
- If you're having trouble, or something isn't working as you think it should − let me know: send the .blend file and a problem description to =nils at thuerey dot de=. Please check these wiki pages and the blenderartists−forum before sending a mail!

Limitations & Workarounds

- One domain per blender file (as of Version 2.4.2), but you can have multiple fluid objects. Workaround: For prelims, move the domain around to encompass each fluid flow part, and then for final, scale up the size of the domain to include all fluid objects (but computation will take longer). This is actually a benefit, because it lets you control how much compute time is used by varying the size and location of the domain.
- If the setup seems to go wrong make sure all the normals are correct (hence enter edit mode, select all, recalculate normals once in a while).

- Currently there's a problem with zero gravity simulation − simply select a very small gravity until this is fixed.
- If an object is inited as volume, it has to be closed, and have an inner side (a plane wont work). To use planes, switch to `Init Shell`, or extrude the plane.
- Blender freezes after clicking `BAKE`. Pressing Esc makes it work again after a while − this can happen if the resolution is too high and memory is swapped to hard disk, making everything horribly slow. Reducing the resolution should help in this case.
- Blender crashes after clicking `BAKE` − this can happen if the resolution is really high and more than 2GB are allocated, causing Blender to crash. Reduce the resolution.
- The meshes should be closed, so if some parts of e.g. a fluid object are not initialized as fluid in the simulation check that all parts of connected vertices are closed meshes. Unfortunately, the Suzanne (monkey) mesh in Blender is not a closed mesh (the eyes are separate).
- If the fluid simulation exits with an error message (e.g. that the init has failed), make sure you have valid settings for the domain object, e.g. by resetting them to the defaults.
- To import a single fluid surface mesh you can use this script: .bobj.−Import−Script.
- You may not be able to bake a fluid that takes more than 1GB, not even with the LargeAddressAware build − it might be a limitation of the current fluid engine.
- Note that first frame may well take only a few hundred MBs of RAM memory, but latter ones go over one GB, which may be why your bake fails after awhile. If so, try to bake one frame at the middle or end at full res so you'll see if it works.
- Memory used doubles when you set surface subdivision from 1 to 2.
- Using "generate particles" will also add memory requirements, as they increase surface area and complexity. Ordinary fluid−sim generated particles probably eat less memory.

See Also

- Tutorial 1: Very Basic Introduction
- Tutorial 2: The Next Step
- Tutorial 1&2 Gui Changes for newer builds
- Developer documentation (implementation, dependencies,...)

External Documentation

- Fluid Simulator Tutorial (video) (Blendernation link) − Very easy to understand video−tutorial to fluid simulation newcomers. Also covers some of the most common pitfalls.
- Guide on Blender Fluid Simulator's Parameters (Blendernation link)

Acknowledgements

The integration of the fluid simulator was done as a Google Summer−of−Code project. More information about the solver can be found at www.ntoken.com. These Animations were created with the solver before its integration into blender: Adaptive Grids, Interactive Animations. Thanks to Chris Want for organizing the Blender−SoC projects, and to Jonathan Merrit for mentoring this one! And of course thanks to Google for starting the whole thing... SoC progress updates were posted here: SoC−Blenderfluid−Blog at PlanetSoC.

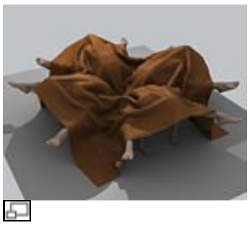
The solver itself was developed by Nils Thuerey with the help and supervision of the following people: U. Ruede, T. Pohl, C. Koerner, M. Thies, M. Oechsner and T. Hofmann at the Department of Computer Science 10 (System Simulation, LSS) in Erlangen, Germany.

http://www10.informatik.uni−erlangen.de/~sinithue/img/lsslogo.png
http://www10.informatik.uni−erlangen.de/~sinithue/img/unierlangenlogo.png

Previous: Manual/Rigid Bodies	Contents	*Next:* Manual/Cloth

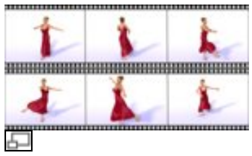
User Manual: Contents | Guidelines | Blender Version **2.46**

Cloth example



Cloth on carved wooden men (made by motorsep)

Cloth example



Cloth example

Cloth Simulation

Cloth simulation is one of the hardest aspects of CG, because it is a deceptively simple real−world item that is taken for granted, yet actually has very complex internal and environmental interactions. After years of development, Blender has a very robust cloth simulator that is used to make clothing, flags, banners, and so on. Cloth interacts with and is affected by other moving objects, the wind and other forces, as well as a general aerodynamic model, all of which is under your control.

Description

Mode: Object Mode

Panel: Editing context&arr;Modifiers panel

Hotkey: F7 to get to Object context; repeat to change sub−context.

A piece of cloth is any mesh, open or enclosed, that has been designated as cloth. The Cloth panels are located in the Physics subcontext and consist of three panels of options. Cloth is either an open or closed mesh and is mass−less, in that all cloth is assumed to have the same density, or mass, per square unit.

Cloth is commonly modeled as a mesh grid primitive, or a cube, but can also be, for example, a teddy bear. However, Blender's Softbody system provides better simulation of closed meshes; Cloth is a specialized simulation of fabrics.

Cloth is a modifier in the object's modifier stack, so it can interact with other modifiers, such as Armature and Smooth. In these cases, the ultimate shape of the mesh is computed in accordance with the order of the modifier stack. For example, you should smooth the cloth AFTER the modifier computes the shape of the cloth.

You can *Apply* the cloth modifier to freeze, or lock in, the shape of the mesh at that frame, which removes the modifier. For example, you can drape a flat cloth over a table, let the simulation run, and then apply the modifier. In this sense, you are using the simulator to save yourself a lot of modeling time.

Results of the simulation are saved in a cache, so that the shape of the mesh, once calculated for a frame in an animation, does not have to be recomputed again. If changes to the simulation are made, the user has full control over clearing the cache and re−running the simulation. Running the simulation for the first time is fully automatic and no baking or separate step interrupts the workflow.

Computation of the shape of the cloth at every frame is automatic and done in the background; thus you can continue working while the simulation is computed. However it is CPU−intensive and depending on the

power of your PC and the complexity of the simulation, the amount of CPU needed to compute the mesh varies as does the lag you might notice.

Don't jump ahead

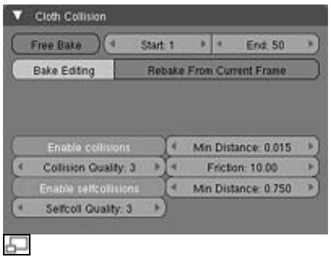
If you set up a cloth simulation but Blender has not computed the shapes for the duration of the simulation, and if you jump ahead a lot of frames forward in your animation, the cloth simulator may not be able to compute or show you an accurate mesh shape for that frame, if it has not previously computed the shape for the previous frame(s).

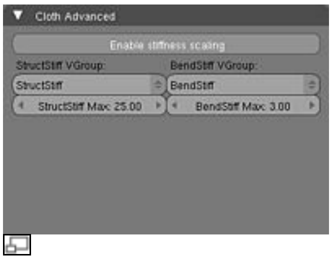
Workflow

A general process for working with cloth is to:

1. Model the cloth object as a general starting shape.
2. Designate the object as a 'cloth' in the 'physics' subcontext of the 'Object' (F7) button window.
3. Model other deflection objects that will interact with the cloth. Ensure the Deflection modifier is last on the modifier stack, after any other mesh deforming modifiers.
4. Light the cloth and assign materials and textures, UV−unwrapping if desired.
5. If desired, give the object particles, such as steam coming off the surface.
6. Run the simulation and adjust Options to obtain satisfactory results. The timeline window's VCR controls are great for this step.
7. Optionally age the the mesh to some point in the simulation to obtain a new default starting shape.
8. Make minor edits to the mesh on a frame−by−frame basis to correct minor tears.

Cloth Panel Options





Cloth panel Cloth collisions panel Advanced cloth functions panel

Cloth is controlled by the three panels shown above. Each of the options (buttons and controls) on those three panels is fully documented in the Reference manual. This section discusses how to use those options to get the effect you want.

Cloth Definition

First, enable *Cloth*. Set up for the kind of cloth you are simulating. You can choose one of the presets to have a starting point.

As you can see, the heavier the fabric, the more stiff it is and the less it stretches and is affected by the air.

Next, position and pin the cloth in the desired position. Pinning is described below. Note that if you move the cloth object *after* you have already run some simulations, you must unprotect and clear the cache; otherwise, Blender will use the position of the current/cached mesh's verticies when trying to represent where they are.

Editing the shape of the mesh, after simulation, is also discussed below. You may disable the cloth and edit the mesh as a normal mesh editing process. −−this is jumping ahead and not clear and not true at this point. −−Roger 18:42, 27 April 2008 (UTC) Finally, use the timeline Play button, or press Alt A in the 3D View to run the simulation. Your cloth will fall and interact with Deflection objects as it would in the real world. −−this is jumping ahead and not clear and not true at this point. −−Roger 18:42, 27 April 2008 (UTC)

Using Simulation to Shape/Sculpt a Mesh

You can Apply the Cloth Modifer at any point to freeze the mesh in position at that frame. You can then re−enable cloth, setting the start and end frames from which to run the simulation forward.

Another example of aging is a flag. Define the flag as a simple grid shape and pin the edge against the flagpole. Simulate for 50 frames or so, and the flag will drop to its "rest" position. Apply the Cloth modifier. If you want the flag to flap or otherwise move in the scene, re−enable it for the framerange when it is in camera view.

Collisions

In most cases, a piece of cloth does not just hang there in 3D space, it collides with other objects in the environment. To ensure proper simulation,there are several items that have to be set up and working together:

1. . The Cloth object must be told to participate in Collisions
2. . Optionally (but recommended) tell the cloth to collide with itself
3. . Other objects must be visible to the Cloth object via shared layers
4. . The other objects must be mesh objects
5. . The other objects may move or be themselves deformed by other objects (like an armature or shape key).
6. . The other mesh objects must be told to deflect the cloth object
7. . The blend file must be saved in a directory so that simulation results can be saved.
8. . You then Bake the simulation. The simulator computes the shape of the cloth for a frame range.
9. . You can then edit the simulation results, or make adjustments to the cloth mesh, at specific frames.
10. . You can make adjustments to the environment or deforming objects, and then re−run the cloth simulation from the current frame forward.

Cloth Collision

Cloth collisions panel

Now you must tell the Cloth object that you want it to participate in collisions. For the cloth object, locate the *Cloth Collision* panel, shown to the right:

Enable Collisions
: LMB click this to tell the cloth object that it needs to move out of the way

Min Distance
: As another object gets this close to it (in Blender Units), the simulation will start to push the cloth out of the way.

Collision Quality
: A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the cloth.

Friction
: A coefficient for how slippery the cloth is when it collides with the mesh object. For example, silk has a lower coefficient of friction than cotton.

Self−collisions

Real cloth cannot permeate itself, so you normally want the cloth to self−collide. ,

Enable Selfcollisions

LMB click this to tell the cloth object that it should not penetrate itself. This adds to simulation compute time, but provides more realistic results. A flag, viewed from a distance does not need this enabled, but a close−up of a cape or blouse on a character should have this enabled.

Min distance

If you encounter problems, you could also change the *Min Dist* value for the self−collisions. The best value is 0.75, for fast things you better take 1.0. The value 0.5 is quite risky (most likely many penetrations) but also gives some speedup.

Selfcoll Quality

For higher self−collision quality just increase the *Selfcoll Quality* and more selfcollision layers can be solved. Just keep in mind that you need to have at least the same *Collision Quality* value as the *Selfcoll Quality* value.

Regression blend file: Cloth selfcollisions

Shared Layers

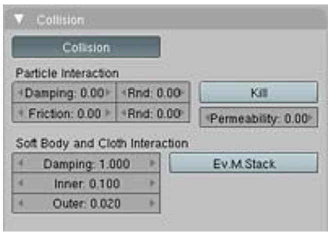
For example, suppose you have two objects: a pair of Pants on layers 2 and 3, and your Character mesh on layers 1 and 2. You have enabled the Pants as cloth as described above. You must now make the Character 'visible' to the Cloth object, so that as your character bends its leg, it will push the cloth. This principle is the same for all simulations; simulations only interact with objects on a shared layer. In this example, both objects share layer 2.

To view/change an object's layers, RMB click to select the object in object mode in the 3D view. M to bring up the Move Layers popup. This popup shows you all the layers that the object is on. To put the object on a single layer, LMB click the layer button. To put the object on multiple layers, Shift LMB the layer buttons. To remove an object from a selected layer, simply Shift LMB the layer button again to toggle it.

Mesh Objects Collide

If your colliding object is not a mesh object, such as a NURBS surface, or text object, you must convert it to a mesh object. To do so, select the object in object mode, and in the 3D View header, select Object−>convert Object type Alt C and select Mesh from the popup menu.

Cloth − Object collisions



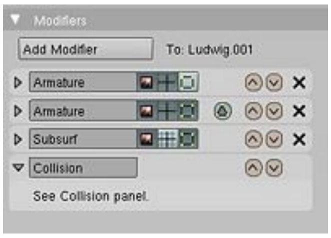
Collision settings

The cloth object needs to be deflected by some other object. To deflect a cloth, the object must be enabled as an object that collides with the cloth object. To enable Cloth − Object collisions, you have to enable deflections on the collision object (not on the cloth object).

In the Buttons Window, Object Context, Physics subcontext buttons, locate the Collision panel shown to the right. It is also important to note that this collision panel is used to tell all simulations that this object is to participate in colliding/deflecting other objects on a shared layer (fluids, soft bodies, and cloth).

Beware: there are two different Collision panels. One is by default a tab beside the Cloth panel. The other is a tab beside the Fields panel. Both are found in the *Physics buttons* Object subcontext. The Collision panel needed here is the latter one.

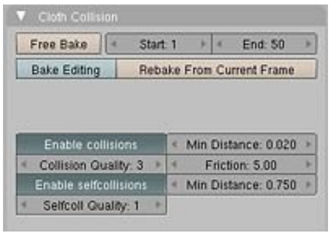
Mesh Object Modifier Stack



Collision stack

The object's shape deforms the cloth, so the cloth simulation must know the "true" shape of that mesh object at that frame. This true shape is the basis shape as modified by shape keys or armatures. Therefore, the Collision modifier must be **after** any of those. The image to the right shows the Modifiers panel for the Character mesh object (not the cloth object).

Bake Collision



After you have set up the deflection mesh for the frame range you intend to run the simulation (including animating that mesh via armatures), you can now tell the cloth simulation to compute (and avoid) collisions. Select the cloth object and in the Object Physics buttons, set the Start: and End: panels for the simulation frames you wish to compute, and click the *Bake* button.

Start
: the starting frame of the animation when you want the cloth to start responding

End
: the end frame of the simulation. The cloth will remain "frozen" after End

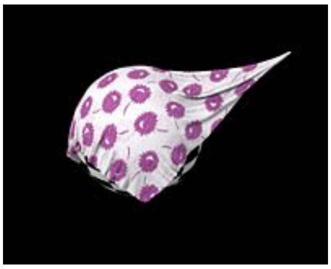
Bake
: Starts the simulation process.

You cannot change Start or End without clearing the bake simulation. When the simuulation has finished, you will notice you have the option to free the bake, edit the bake and re−bake.

Free Bake
: Deletes the simulation, enabling you to make changes and then start over

Bake Editing
: Enables you to protect the cache and prevent the simulation from recomputing the mesh shape. You can then edit the mesh shape manually.

Rebake From Current Frame
: Enables you to re−compute the cloth simulation from the current frame to End:



Standard settings cloth collide with sphere

There's a few things you'll probably notice right away. First, it will bake significantly slower than before, and it will probably clip through the box pretty bad as in the picture on the right.

Editing the cached simulation

The cache contains the shape of the mesh at each frame. You can edit the cached simulation, when you baked the simulation and pressed the *Bake Editing* button. Just go to the frame you want to fix and TAB into

editmode. There you can move your vertices using all of Blender's mesh shaping tools. When you exit, the shape of the mesh will be recorded for that frame of the animation. If you want Blender to resume the simulation using the new shape forward, LMB click *Rebake from next Frame* and play the animation. Blender will then pick up with that shape and resume the simulation.

Edit the mesh to correct minor tears and places where the colliding object has punctured the cloth

If you add, delete, extrude, or remove the vertices in the mesh, Blender will take the new mesh as the starting shape of the mesh back to the *First Frame* of the animation, replacing the original shape you started with, up to the frame you were on when you edited the mesh. Therefore, if you change the content of a mesh, when you Tab out of edit mode, you should unprotect and clear the cache *From next frame* so that Blender will make a consistent simulation.

Troubleshooting

If you encounter some problems with collision detection there are two ways to fix them:

- The fastest solution would be to put up the 'Min Distance' setting under the Cloth Collisions panel. This will be the fastest way fix the clipping, however, it will be less accurate and won't look as good. Using this method tends to make it look like the cloth is resting on air, and gives it a very rounded look.

- A second method is to increase the "Quality" (in the first panel). This results in smaler steps for the simulator and therefore to a higher propability that fast moving collisions get catched. You can also increase the "Collision quality" to perform more iterations to get collisions solved.

- If none of the methods help, you can easily edit the cached/baked result in editmode afterwards.

My Cloth is torn by the deforming mesh − he "Hulks Out"
: Increase the Structural Stiffness very high, like 1000

Subsurf modifier

A bake/cache is done for every subsurf level so please use ONE EQUAL subsurf level for render and preview.

Examples

To start with cloth, the first thing you need, of course, is some fabric. So, lets delete the default cube and add a plane. I scaled mine up along the Y axis, but you don't have to do this. In order to get some good floppy and flexible fabric, you'll need to subdivide it several times. I did it 8 times for this example. So **TAB** into edit mode, and press **WKEY** and then 'Subdivide multi', and set it to 8. Now, we'll make this cloth by going to the Object buttons(F7), then the sub−section Physics buttons. Scroll down until you see the 'Cloth' panel, and press the Cloth button. Now, a lot of settings will appear, most of which we'll ignore now.

That's all you need to do to set your cloth up for animating, but if you hit Alt A, your lovely fabric will just drop very un−spectacularly. That's what we'll cover in the next two sections about pinning and colliding.

Pinning of cloth

Cloth in action

The first thing you need when pinning cloth are Vertex Groups. There are several ways of doing this including using the Weight Paint tool to paint the areas you want to pin (see the Weight paint section of the manual).

Once you have a vertex group set, things are pretty straightforward, all you have to do is press the 'Pinning of cloth' button in the Cloth panel and select which vertex group you want to use, and the stiffness you want it at. You can leave the stiffness as it is, default value is fine.

Smoothing of Cloth

Now, if you followed this from the previous section, your cloth is probably looking a little blocky. In order to make it look nice and smooth like the picture you need to apply a Smooth and/or Subsurf modifier in the modifier panel under the Editing buttons(F9). Then, also under the editing buttons, find the Links and Materials panel (the same one you used for vertex groups) and press 'Set Smooth'.

Now, if you hit Alt A Things are starting to look pretty nice, don't you think?

Cloth on armature

Cloth deformed by armature and also respecting an additional collision object: Regression blend file

Using Cloth for Softbodies

Using cloth for softbodies

Cloth can also be used to simulate softbodies. It's for sure not it's main purpose but it works nontheless. The example image uses standard "rubber" material, no fancy settings, just ctrl−a.

Blend file for the example image: Using Cloth for softbodies

Cloth with Wind

Flag with wind applied

Regression blend file for Cloth with wind and selfcollisions (also the blend for the image above): Cloth flag with wind and selfcollisions

Reference

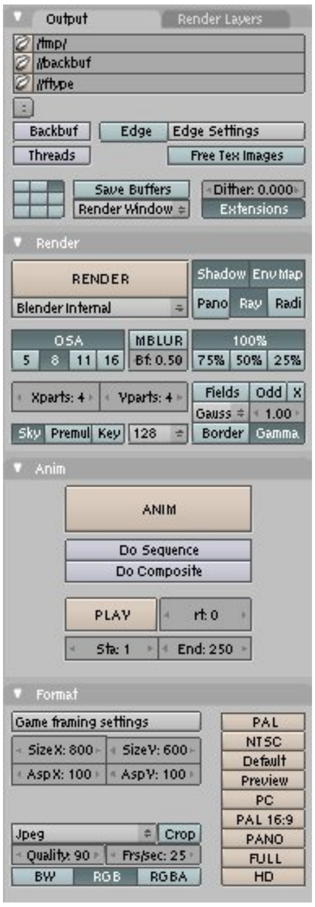
For more information, consult the developer's Main development page of Blender Cloth.

Previous: Manual/Fluid Simulation Contents *Next:* Manual/Rendering

User Manual: Contents | Guidelines | Blender Version **2.45**

Rendering is the final process of CG (short of post processing, of course) and is the phase in which the image corresponding to your 3D scene is finally created. Rendering is a CPU−intensive process. You can render an image on your computer, or use a render farm which is a network of PC's that each work on a different section of the image or on different frames. This section provides a full explanation of the features in Blender related to the process of producing your image or animation.

Introduction



Rendering Buttons.

The rendering buttons window is accessed via the Scene Context and Render Sub−context (F10 or the  button). The rendering Panels and Buttons are shown in *Rendering Buttons.*.

These buttons are organized into panels, which are:

- Output − controls the output of the render pipeline
- Render Layers − controls which layers and passes to render
- Render − controls the actual rendering process of a still shot
- Anim − controls the rendering of a series of frames to produce an animation
- Bake − pre−computes certain aspects of a render
- Format − controls the format and encoding of the picture or animation
- Stamp − stamps the frames with identifying and configuration control item information

Tabs

To save screen space, some of the panels may be tabbed under another; for example, the Layers panel is a tab folder under the Output panel. To reveal it, simply click the tab header.

Yafray

If you have installed Yafray, options to control it will appear as two tabs under the Render panel once you have selected it as a rendering engine

Overview

The rendering of the current scene is performed by pressing the big RENDER button in the Render panel, or by pressing F12 (you can define how the rendered image is displayed on−screen in the Render Output Options). See also the Render Window.

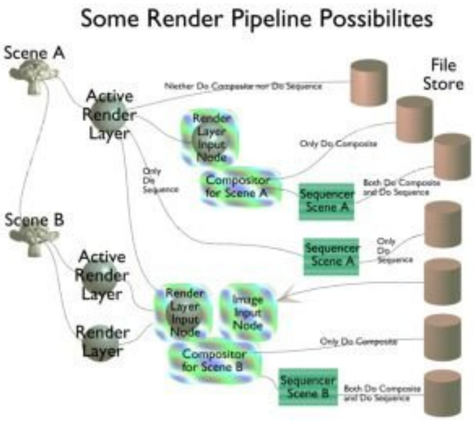
A movie is produced by pressing the big ANIM animation button in the Anim panel. The result of a rendering is kept in a buffer and shown in its own window. It can be saved by pressing F3 or via the File−>Save Image menu using the output option in the Output panel. Animations are saved according to the format specified, usually as a series of frames in the output directory. The image is rendered according to the dimensions defined in the Format Panel (*Image types and dimensions.*).

Workflow

In general, the process for rendering is:

1. Create all the objects in the scene
2. Light the scene and position the camera
3. Render a test image at 25% or so without oversampling or raytracing etc. so that it is very fast and does not slow you down
4. Set and Adjust the materials/textures and lighting
5. Iterate the above steps until satisfied at some level of quality
6. Render progressively higher−quality full−size images, making small refinements and using more compute time
7. Save your images

Render Workbench Integration



Blender has three independent rendering workbenches which flow the image processing in a pipeline from one to the other in order:

- Rendering Engine
- Compositor
- Sequencer

You can use each one of these independently, or in a linked workflow. For example, you can use the Sequencer by itself to do post processing on a video stream. You can use the Compositor by itself to perform some color adjustment on an image. You can render the scene, via the active Render Layer, and save that image directly, with the scene image computed in accordance with the active render layer, without using the Compositor or Sequencer. These possibilities are shown in the top part of the image to the right.

You can also link scenes and renders in Blender as shown, either directly or through intermediate file storage. Each scene can have multiple render layers, and each Render Layer is mixed inside the Compositor. The active render layer is the render layer that is displayed and checked active. If the displayed render layer is not checked active/enabled, then the next checked render layer in the list is used to compute the image. The image is displayed as the final render if Do Composite and Do Sequence are NOT enabled.

If Do Composite is enabled, the render layers are fed into the Compositor. The noodles manipulate the image and send it to the Composite output, where it can be saved, or, if *Do Sequence* is on, it is sent to the Sequencer.

If Do Sequence is enabled, the result from the compositor (if Do Composite is enabled) or the active Render layer (if Do Composite is not enabled) is fed into the Scene strip in the Sequencer. There is is manipulated according to the VSE settings, and finally delivered as the image for that scene.

Things get a little more complicated when a .blend file has multiple scenes, for example Scene A and Scene B. In Scene B, if Do Composite is enabled, the Render Layer node in Scene B's compositor can pull in a Render Layer from Scene A. Note that this image will not be the post−processed one. If you want to pull in the composited and/or sequenced result from Scene A, you will have to render Scene A out to a file using Scene A's compositor and/or sequencer, and then use the Image input node in Scene B's compositor to pull it in.

The bottom part of the possibilities graphic shows the ultimate blender: post−processed images and a dynamic component render layer from Scene A are mixed with two render layers from Scene B in the compositor, then sequenced and finally saved for your viewing enjoyment.

These examples are only a small part of the possibilities in using Blender. Please read on to learn about all the options, and then exercise your creativity in developing your own unique workflow.

Sequencer from/to Compositor

To go from the Compositor to the Sequencer, enable both "Do Sequence" and "Do Composite". In the Compositor, the image that is threaded to the Composite Output node is the image that will be processed in the Scene strip in the VSE.

The way to go from the Sequencer to the Compositor is through a file store. Have one scene "Do Sequence" and output to an image sequence or mov/avi file. Then, in another scene, "Do Composite" and using an image input node to read in the image sequence or mov/avi file.

Render Window Options

Once you press F12 or click the big Render button, your image is computed and display begins. Depending on the Output Panel Option, the image is shown in a separate Render Window, Full Screen, or in a UV/Image Editor window.

You can render a single frame, or many frames in an animation. You can cancel the rendering by pressing Esc. If rendering a sequence of frames, the lowest number frame is rendered, then the next, and so on in increasing frame number sequence.

The Render Window can be invoked in several ways:

Render Panel−>Render or F12
: renders the current frame, as seen by the active camera, using the selected renderer (Blender Internal or Yafray)

3D View Window Header−> LMB OpenGL Render button (far right)
: Renders a quick preview of the 3D View window

Anim Panel−>Anim Ctrl F12
: Render the Animation from the frame start to and included in the End frame, as set in the Anim panel.

3D View Window Header−>Ctrl LMB OpenGL Render button (far right)
: Renders a quick animation of the 3D View window using the fast OpenGL render

Output Options needs to be set correctly. In the case of Animations, each frame is written out in the Format specified.

Rendering the 3D View Animation using the OpenGL is useful for showing armature animations.

Showing Previous Renders

If the Blender Internal render was used to compute the image, you can look at the previous render:

Render−>Show Render Buffer
: F11 − Pops up the Render Window and shows the last rendered image (even if it was in a previously opened & rendered .blend file).

Render−>Play Back Rendered Animation
: Ctrl F11 − Similar as for the single frame, but instead plays back all frames of the rendered animation.

Render Window usage

Once rendering is complete and the render window is active, you can:

- A − Show/hide the alpha layer.

- Z − Zoom in and out. Pan with the mouse. You can also mousewheel to zoom

- J − Jump to other Render buffer. This allows you to keep more than one render in the render window, which is very useful for comparing subtleties in your renders. How to use it:

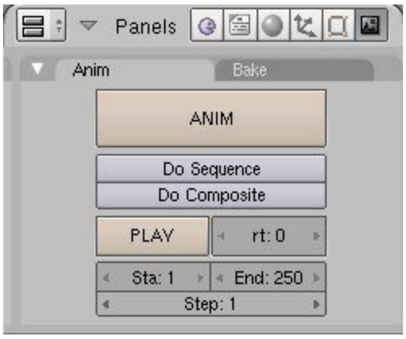
1. Press Render or F12
2. Press J to show the empty buffer (the one we want to "fill" with the new image)
3. Go back to the Blender modeling window. You can send the render window to the background by pressing Esc. *Do not* close, or minimize the render window!

4. Make your changes.
5. Render again.
6. Press J to switch between the two renderings.

* LMB − Clicking the left mouse button in the Render window displays information on the pixel underneath the mouse pointer. The information is shown in a text area at the bottom left of the Render output window. Moving the mouse while holding the LMB will dynamically update the information. This information includes:
    * Red, Green, Blue and Alpha values
    * Distance in Blender Units from the camera to the pixel under the cursor (Render Window only)

For Alpha values to be computed, RGBA has to be enabled. Z−depth (distance from camera) if computed only if Z is enabled on the Render Layers tab.

Step Render Frame



Step Frame Option.

Blender allow you to do faster animation renders skipping some Frames. You can set the step in the Render Panel as you can see in the picture.

Once you have your video file rendered, you can play it back in the real speed (fps). For that you just need to set the same Step parameter you used to render it.

If you play the stepped video in a normal speed in an external player, the general speed will be *Step* times faster than a video rendered with all the frames (eg. a video with 100 frames rendered with Step 4 @ 25 fps will have only 1 second. The same video rendered with Step 1 (the default value) will have 4 seconds of length).

If you want to use this parameter for rendering through command line you can use −j STEP, where STEP stands for the number of steps you want to use.

If you want to use this parameter for playing video files through the command line, you need to use the parameter −j STEP following the −a (which stands for the playback mode).

./blender −a −s 1 −e 100 −p 0 0 −f 25 1 −j 4 "//videos/0001.jpg"

- Rendered stepped frames with output video format such as FFMpeg (always) or AVI Jpeg (sometimes) produce a corrupted video (missing end frames). in Blender 2.48a. Therefore the rendered video can't be played−back properly.

Therefore I suggest to work with the video output format of JPG, it works fine all the time.

Distributed Render Farm

There are several levels of CPU allocation that you can use to decrease overall render time by applying more brainpower to the task.

First, if you have multi−core CPU, you can increase the number of threads, and Blender will use that number of CPUs to compute the render.

Second, if you have a local area network with available PC's, you can split the work up by frames. For example, if you want to render a 200 frame animation, and have 5 PC's of roughly equal processing power, you can allocate PC#1 to produce frames 1−40, PC#2 to frames 41−80, and so on. If one PC is slower than the others, simply allocate fewer frames to that PC. To do LAN renders, map the folder containing the .blend file (in witch you should have packed your external datas, like the textures, â€¦) as a shareable drive. Start Blender on each PC and open the .blend file. Change the Start and End frame counts on that PC, but do not save the .blend file. Start Rendering. If you use relative paths for your output pathspec, the rendered frames will be placed on the host PC.

Third, you can do WAN rendering, which is where you email or fileshare or Verse−share the .blend file (with packed datas!) across the Internet, and use anyone's PC to perform some of the rendering. They would in turn email you the finished frames as they are done. If you have reliable friends, this is a way for you to work together.

Fourth, you can use a render farm service. This service, like BURP, is run by an organization. You email them your file, and then they distribute it out across their PC's for rendering. BURP is mentioned because it is free, and is a service that uses fellow BlenderHead PC's with a BOINC−type of background processing. Other services are paid subscription or pay−as−you−go services.

Previous: Manual/Fluid Simulation	Contents	*Next:* Manual/Output Options

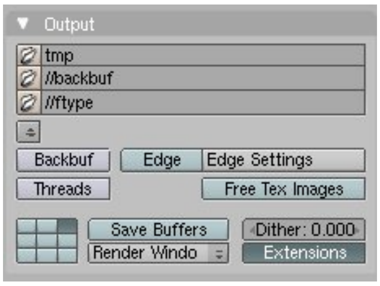
User Manual: Contents | Guidelines | Blender Version **2.43**

Output Panel

Mode: All Modes

Panel: Render Context Render

Hotkey: F10



Render Output buttons.

This panel provides many options for rendering, increasing and optimizing your render and output speed, and the location for displaying and saving your render output. The options on this panel control where and how the results of a render are handled.

File Locations

At the top are three filespec fields:

- Animation Output Directory and filename prefix (default: //tmp)
- Background image (default: //backbuf)
- FType image format descriptor file (default: //ftype)

By default, each frame of an animation is saved in the directory where the file was loaded from (the "//" part of the path specification) and given a filename that begins with "tmp". Change this or any field by Shift LMB clicking in the name field and entering a new name. If you use the // and do not save a new .blend file somewhere, Blender assumes the // to refer to the Blender install folder.

A background image, such as a studio curtain, a watermark, or any image may be used as a background. You usually want to set the window background to this picture, and when you render, use the background image in the render output.

Ftype uses an "Ftype" file, to indicate that this file serves as an example for the type of graphics format in which Blender must save images. This method allows you to process 'color map' formats. The color map data is read from the file and used to convert the available 24 or 32 bit graphics. If the option "RGBA" is specified, standard colour number '0' is used as the transparent colour. Blender reads and writes (Amiga) IFF, Targa, (SGI) Iris and CDinteractive (CDi) RLE colormap formats.

Directory Browser

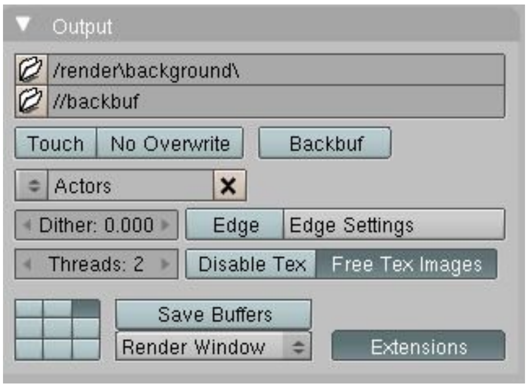
Clicking the folder icon to the left of the field turns a Blender window pane into a File Browser window.

Using this window is very handy for scrolling through your hard disk and selecting a file or directory.

PathSpecs

The path specification for the location can include a normal or mapped drive letter (e.g. "F:"), a breadcrumb notation (e.g. "./" and "../" and "//" (file location). Forward slashes (Unix−style) or backslashes (Windows−style) are acceptable on either platform. If omitted, the file is saved in the Blender Installation directory.

Scene Linking



Background scene linking in Actors Scene

The up−down button just below the ftype field lists the scenes in the file and allows you to link in another scene. When you select a scene, you will note that all the objects in that scene are shown in your 3D windows. They have a special outline and you cannot select them, but they are shown and will participate in the render. The linked scene can in turn link to a third scene, and so on infinitum; you can form a chain, linking scenes together for an ordered set of scenes to all be rendered together. This feature almost makes scenes akin to layers.

Use this feature for rendering complicated scenes, yet retaining the ability to quickly work on and test render only one major set of objects at a time by un−linking. The render is relative to the current scene's camera, so as long as all objects are positioned properly relative to 3D Space

This feature is useful for teamwork. For example, one team can be working on props and background, while another team works on characters. The backgrounds team can, at any time, link to the actor's scene, and make sure that the actors are not walking through tables and walls, etc.

Other Buttons and Options

Backbuf
: Clicking the Backbuff (off by default) makes blender use the image file specified above in the backbuf field as a background image.

Edge
: Edge button enables cartoon−style outlining of edges of objects. When clicked and enabled, The edge settings dynamic menu allows you to set the Edge Intensity (Eint) of the edge and its color via the

RGB sliders (default:black).

Threads:
Threads enables multi−threaded rendering; great for multi−core and multi−processor systems. On an multi−processor PC, this allows the rendering work (a Thread) to be divided among each processor. If you have a dual−core PC (Intel or AMD), set this to 2 (and increase Xparts and Yparts)

Disable Tex
Using textures, either procedural or image, takes a tremendous amount of processing power. To speed up draft renders, enable this button

Free Tex Images
To save memory, enable the Free Tex Images so that texture images are cleaned out of memory.

Render Location button grid
The 3x3 Grid specifies where (generally) the render window should pop up.

Save Buffers
Save Buffers saves the intermediate layers and render pipeline images to disk as an OpenEXR file, in order to free up physical memory. Use this if you are running out of memory; there will be more hard disk cache used and hence rendering may take little longer because of I/O increases, but you will be able to render HUGE images on your PC.

Error: Sorry...exr tile saving only allowed with equally sized parts:

The XParts and YParts must divide the image up into square patches with respect to pixel size. For example, a 1920x1080 HD image is in a 16:9 ratio; 120 times 16:9 gives 1920x1080. So, an XParts of 16 and YParts of 9 would divy up the render into 144 patches that are 120 pixels square.

Other combinations for XParts:YParts for any sized image in a 16:9 aspect ratio are 64:36 (2304 patches), 80:45 (3600 patches), and 96:54 for over 5000 patches. Using XParts of 80 and YParts of 45, you can render a 3 million polygon image at 50% of HD using full textures and very complicated lighting (ED Scene 7 Shot 4) on a PC with 2G RAM.

Render Display
Render Display is a dynamic menu that lets you select where the image should be displayed
◊ Render Window (default) − In a pop−up window.
◊ Full Screen − Full−screen on your monitor.
◊ Image Editor − Piped directly into Blender's built−in Image Editor.
When rendering to an Image Editor window or a popup Render Window, the grid of 9 buttons allows you to select the location of the window, either the window frame that will be used to become the Image Editor window, or the general location of the popup Render Window when it pops up.

Dither: setting
Dither numeric field (0 is off) specifies the amount of noise to inject into the picture.

Extensions toggle button
Extensions tells blender to add a file extension onto the output file based on the format of the picture. For example, frame 35 of a series of JPEG−encoded pictures would be named "tmp0035.jpg". You almost always want Blender to add the filetype extension so that the image is "recognized" by your operating system.

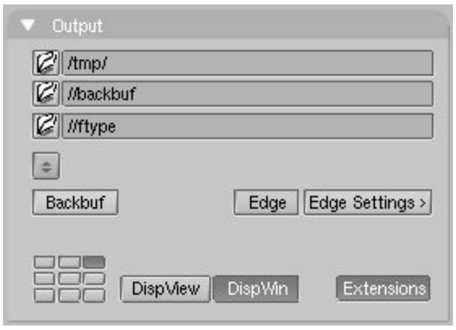
Edge (Toon) Rendering

Description

A scene with Toon materials.

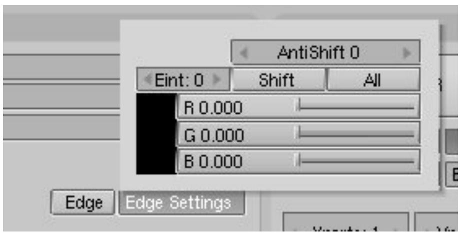
Blender's toon shaders, can give your rendering a comic−book−like or manga−like appearance, affecting the shades of colours, as you may be able to appreciate in *A scene with Toon materials.*. The effect is not perfect since real comics and manga also usually have china ink outlines. Blender can add this feature as a post−processing operation.

Options



Toon edge buttons *(F10)*.

Edge
: This makes Blender search for edges in your rendering and add an 'outline' to them.



Toon edge settings *(F10)*.

Before repeating the rendering it is necessary to set some parameters. The Edge Settings opens a window to set these (*Toon edge settings (F10).*). Edge Settings

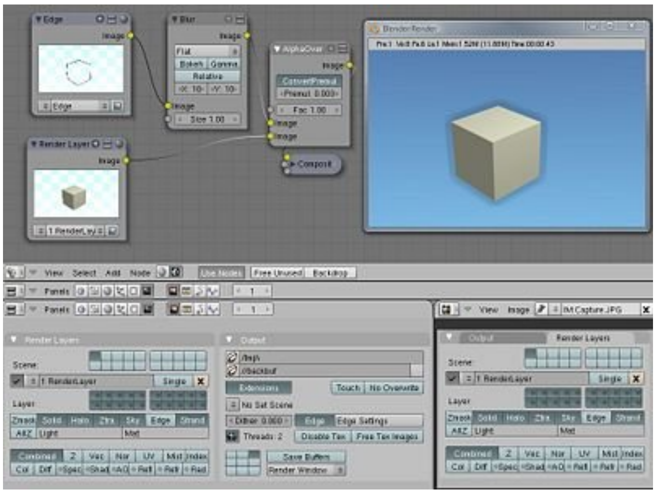
Colour / R/G/B
: The colour of the rendered edges (black by default). Use the sliders or click on the swatch to see the color picker

Eint
: The edge's intensity from 0 to 255. 10 gives outline of object against the background, whereas higher

settings start to pick up forward or leading edges based on the contrast in the image caused by the geometry of the object and not specular spots caused by lighting. At maximum intensity, Edge will even faintly display geometry subsurf edge lines in areas of imperfect smoothing.

Examples

Scene re−rendered with toon edge set.



It is possible to separate out the edge layer using a renderlayer dedicated to that purpose. The alpha channel is 0 where there is no edge, and 1 where the edge is. By separating out the edge layer, you can blur it, change its color, mask it, etc. The image to the right shows how to do this. I created an Edge renderlayer that only has the Sky and Edge layers (I included sky so that we get the world color later on in the composite output). The other renderlayer omits the Edge layer, so it returns just the normal image. On the output panel I enabled Edge with a width of 10 in black. I run that layer through a blur node. Using the Alphaover node, I then composite the cube on top of the blurred edge. The result gives a soft−shadow kind of effect. Note that Premultiply is set because the Edge image already has an alpha of 1.0 set.

Previous: Manual/Rendering	Contents	*Next:* Manual/Oversampling (Antialiasing)

User Manual: Contents | Guidelines | Blender Version **2.43**

Oversampling

Mode: All Modes

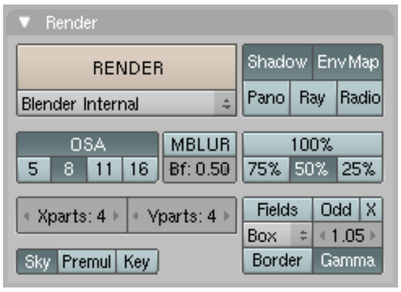
Panel: Render Context Render

Hotkey: F10

Description

A computer generated image is made up of pixels, these pixels can of course only be a single colour. In the rendering process the rendering engine must therefore assign a single colour to each pixel on the basis of what object is shown in that pixel. This often leads to poor results, especially at sharp boundaries, or where thin lines are present, and it is particularly evident for oblique lines. To overcome this problem, which is known as *Aliasing*, it is possible to resort to an Anti−Aliasing technique. Basically, each pixel is 'oversampled', by rendering it as if it were 5 pixels or more, and assigning an 'average' colour to the rendered pixel. The buttons to control Anti−Aliasing, or OverSAmple (OSA), are below the rendering button in the Render Panel (*Render Panel.*).

Options



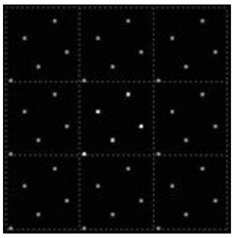
Render Panel.

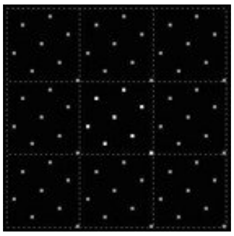
OSA
: Enables oversampling

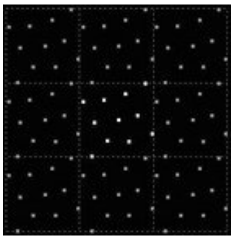
5 / 8 / 11 / 16
: The number of samples to use. The values of OSA (5, 8, 11, 16) are pre−set numbers in specific sample patterns; a higher value produces better edges, but slows down the rendering. By default, we use in Blender a fixed "Distributed Jitter" table. The samples within a pixel are distributed and jittered in a way that guarantees two characteristics:

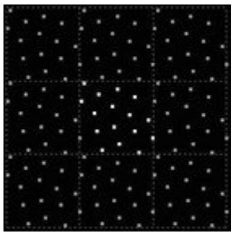
1. Each sample has equal distances to its neighbour samples
2. The samples cover all sub−pixel positions equally, both horizontally and vertically

The images below show Blender sample patterns for 5, 8, 11 and 16 samples. To show that the distribution is equalized over multiple pixels, the neighbour pixel patterns were drawn as well. Note that each pixel has an identical pattern.


5 samples


8 samples


11 samples


16 samples

Filtering

When the samples have been rendered, we've got color and alpha information available per sample. It then is important to define how much each sample contributes to a pixel. The simplest method is to average all samples and make that the pixel color. This is called using a "Box Filter". The disadvantage of this method is that it doesn't take into account that some samples are very close to the edge of a pixel, and therefore could influence the color of the neighbour pixel(s) as well.

Filter menu
: The filter type to use to 'average' the samples:

  Box
  : The original filter used in Blender, relatively low quality. For the Box Filter, you can see that only the samples within the pixel itself are added to the pixel's color. For the other filters, the formula ensures that a certain amount of the sample color gets distributed over the other pixels as well.

  Tent
  : A simplistic filter that gives sharp results

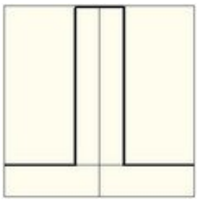
  Quad
  : A Quadratic curve

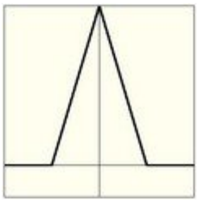
  Cubic
  : A Cubic curve

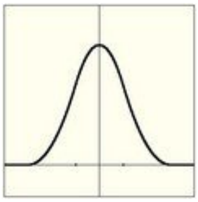
  Gauss
  : Gaussian distribution, the most blurry

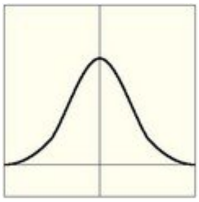
  CatRom
  : Catmull−Rom filter, gives the most sharpening

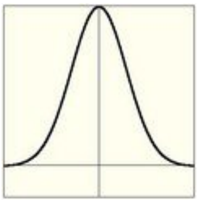
  Mitch
  : Mitchell−Netravali, a good allround filter that gives reasonable sharpness

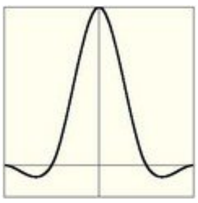

Box

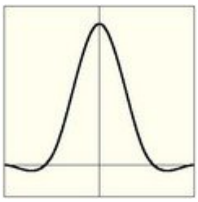

Tent


Quadratic


Cubic

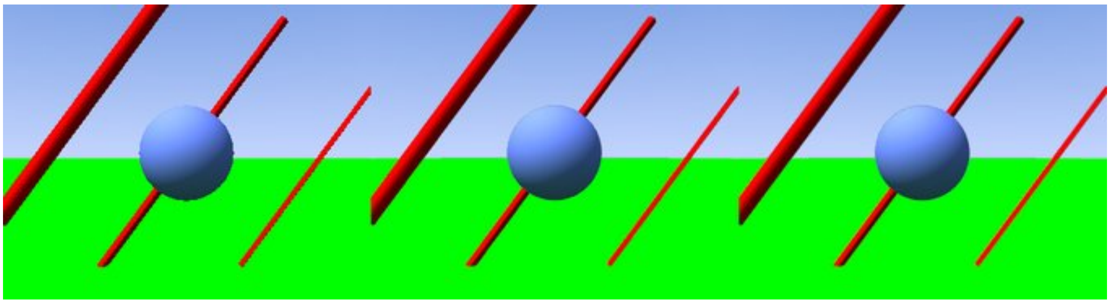

Gaussian


Catmull−Rom

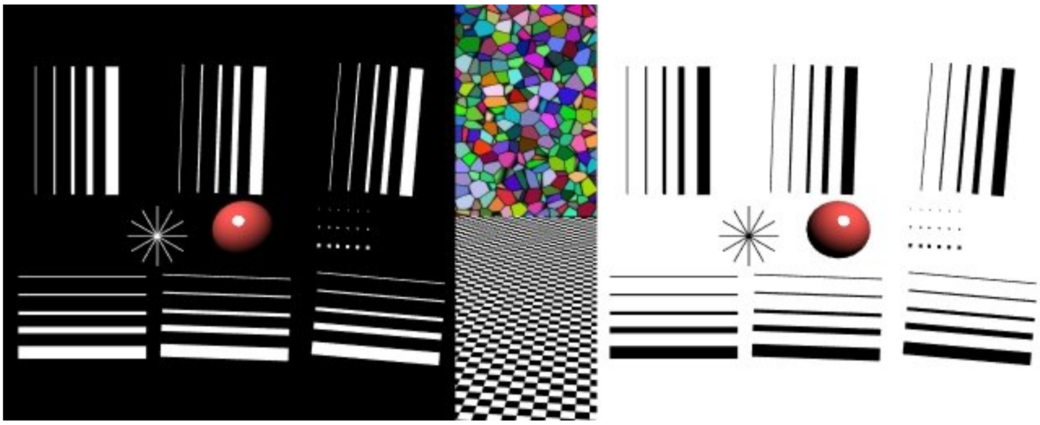

Mitchell−Netravali

Making the filter size value smaller will squeeze the samples more into the center, and blur the image more. A larger filter size make the result sharper. Notice that the last two filters also have a negative part, this will give an extra sharpening result.

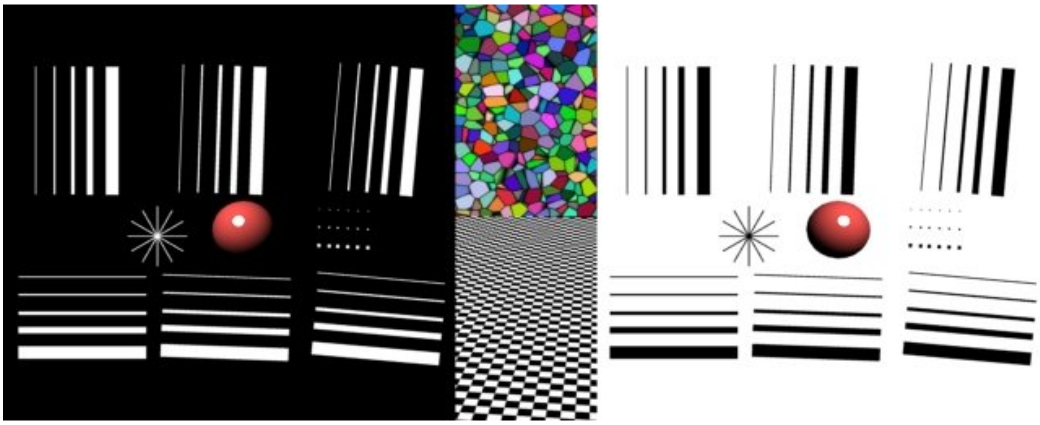
Examples



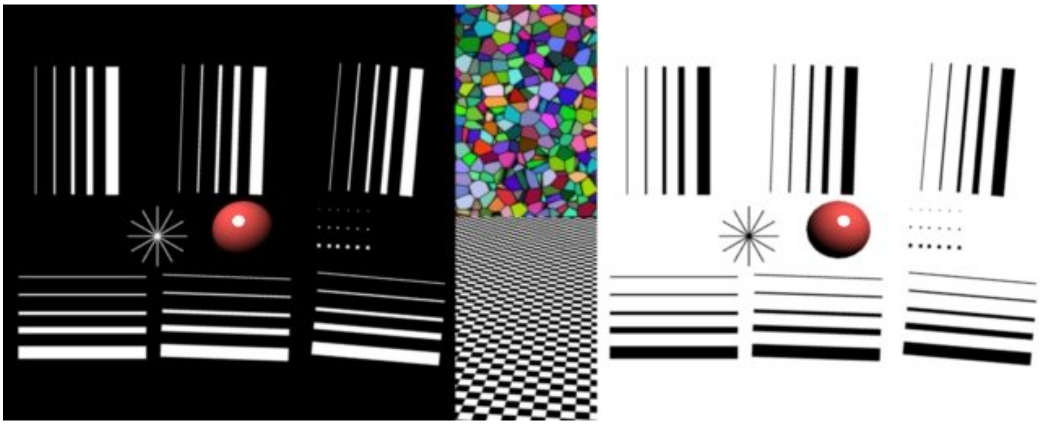
Rendering without OSA (left) with OSA=5 (center) and OSA=8 (right).



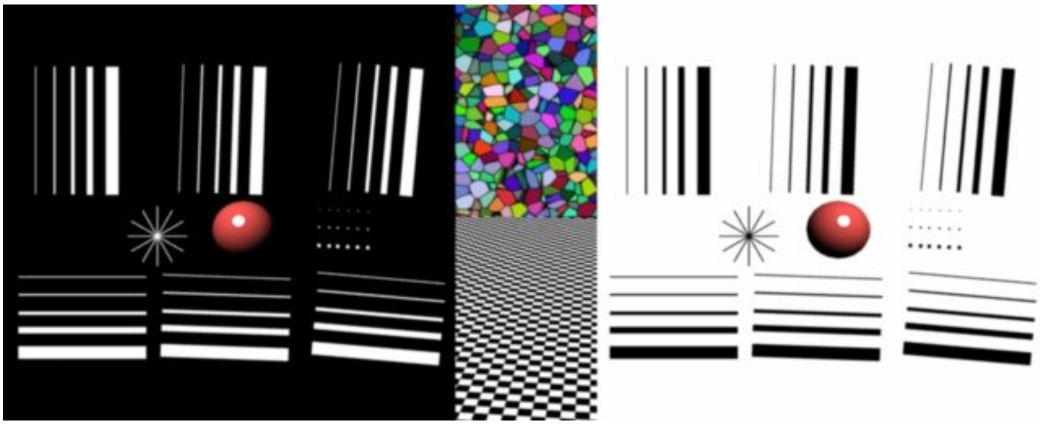
OSA 8, Box filter



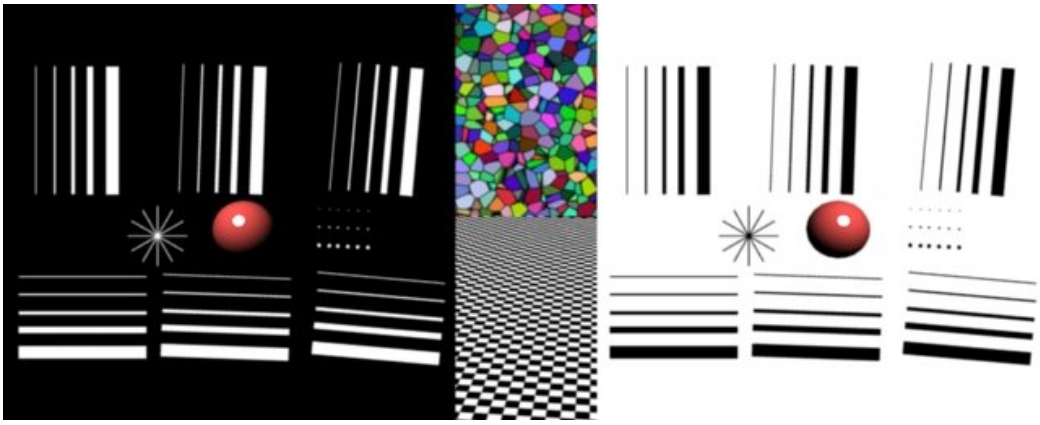
OSA 8, Tent filter



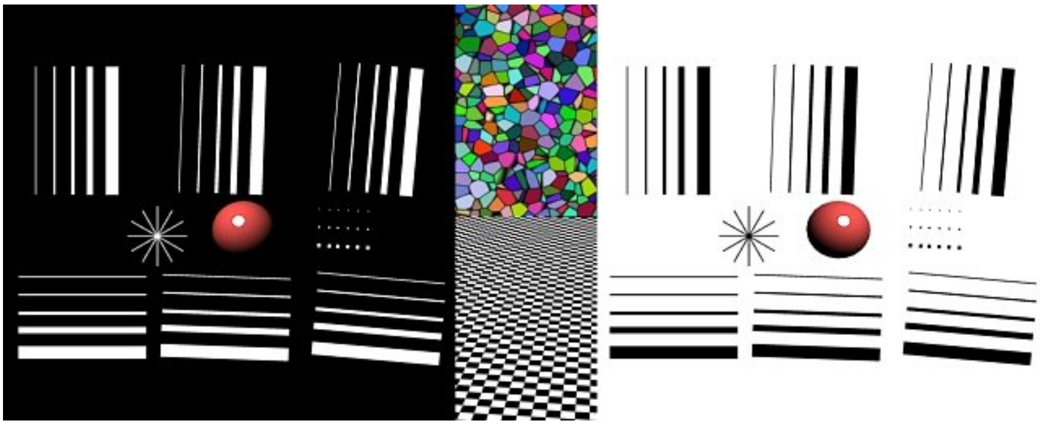
OSA 8, Quadratic filter



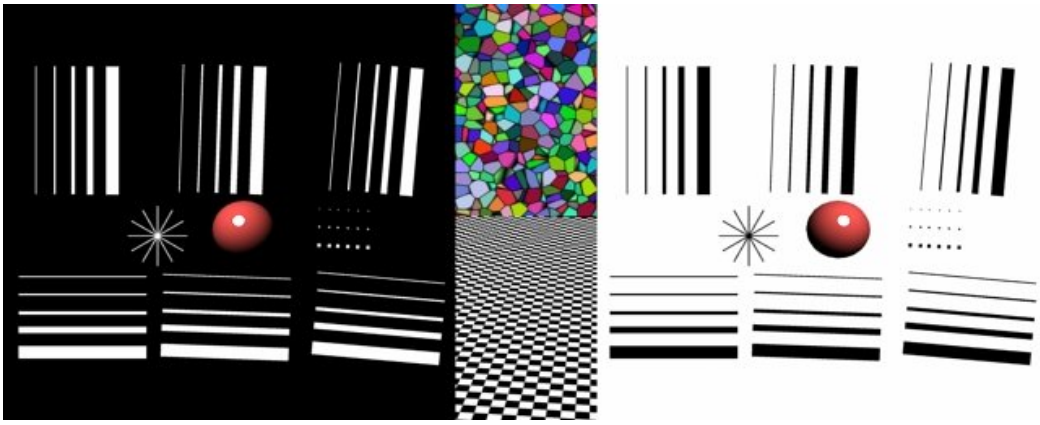
OSA 8, Cubic filter



OSA 8, Gaussian filter



OSA 8, Catmull−Rom filter



OSA 8, Mitchell−Netravali filter

Previous: [Manual/Output Options] [Contents] *Next:* [Manual/Rendering Options]

User Manual: Contents | Guidelines | Blender Version **2.45**

We know that around the world, our users have PC's of widely varying power. Rendering is *the* process in CG that can chew up CPU and disk space like no tomorrow. Especially in corporate environments, it is easy to fill up terabyte servers by uploading ten hour−long DV tapes and doing some editing. So, there are lots of options try to shoehorn a big job into a small PC by providing you with multiple sets of options that chunk up the work as best we can, while still preserving image integrity.

This page discusses the main options found on the Render panel, and subsequent pages give you more.

Shadow

Enable this button to compute and render shadows. Shadows are cast by shadow−casting lights and are received by shadow−able materials. The results of this calculation are made available in the Shadow render pass. For more information on lights, materials, or render passes, please consult the application sections of the wiki.

EnvMap

Environmental mapping changes the background and color of objects based on the light cast by the world environment maps. Enable for more realistic rendering.

Raytracing

Raytracing is a more precise method of computing the color of a surface, especially reflective surfaces. It is enabled by clicking the Ray button.

Octree resolution

Mode: All Modes

Panel: Render Context Render

Hotkey: F10

When raytracing a really big scene with small objects, render times can grow exponentially. To help keep render times down (but use more memory), you can increase the Octree memory allocation. When Raytracing is enabled (Ray button, next to Render button on the Render panel), the Octree select appears at the bottom of the panel, with selectable settings ranging up to 512.

The OctTree is a ray−tracing acceleration structure. It subdivides the whole scene into a regular 3D grid of cells and places every polys of every objects into their corresponding cell. When rendering, it is quicker to determine which cell a ray is intersecting and then test the polys in those cells instead of testing every polys in the scene.

When you have a large low−poly scene with a small high−poly model in the middle, if the octtree resolution is too small, most of the scene's poly that comes from the small high−poly model, end up in a very small

number of cells. And so, even though it is relatively quick to figure which cell to test, when a cell contains a large number of polys, we gained nothing because all those polys must be tested.

On the other hand, if the octtree resolution is too large for a given scene, then the raytracer looses time checking for cells that contains no polys. That is why each scene have its own octtree resolution sweet spot that must be found experimentally.

So, use a high setting for a large open space with small areas that have high−poly structures, and low settings for scenes where the polys (faces) are evenly distributed. For more information, consult the Release Notes for Octree resolution

Radiosity

When light hits an object, some color spectrum of the light is absorbed and other colors are reflected to our eyes. Radiosity is when that light also hits an object close to it, giving color to it, which then in turn is sent to our eyes. Sometimes called color bleeding, because the color of one object bleeds onto the one next to it, radiosity gives much more photorealism.

Percentage Size

Calculating a full size image takes time. For interim renders, Click the 75%, 50% or 25% boxes to render a smaller image (which takes less time). When looking at the render window, you can always expand it and roll your mousewheel to zoom in / expand the image.

Parts Rendering

Mode: All Modes

Panel: Render Context Render

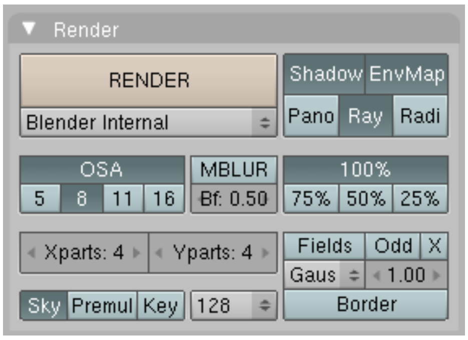
Hotkey: F10

Description

It is possible to render an image in pieces, one after the other, rather than all at one time. This can be useful for very complex scenes, where rendering small sections one after the other only requires computation of a small part of the scene, which uses less memory.

On a multi−core CPU, each part is assigned to a CPU/core, so setting up multiple parts, coupled with increasing the number of threads, speeds up render time by using all of your PC's processing power.

Options



Rendering by parts buttons *(F10)*.

By setting values different from 1 in the Xparts and Yparts NumButtons in the Render Panel (*Rendering by parts buttons.*), you force Blender to divide your image into a grid of Xparts times Yparts sub−images, which are then rendered one after the other and finally assembled together.

Memory is allocated per thread; with each thread doing a full tile render. Almost all geometry (render faces/vertices) is checked for each tile it renders, giving some extra overhead. There's also thread tables for AO and area−shadow lamp sampling, precalculated and allocated in advance. Lastly; there's still quite some locking going on, for each memory allocation call to the operating system (malloc) for example.

A quick rule−of−thumb is to make sure the total number of threads renders less than one quarter of the entire image, to prevent too much memory allocated:

8 threads: use 64 tiles (for example, X and Y Parts = 8) 16 threads: use 128 tiles 128 threads: use 1024 tiles

Whether such giant amount of tiles (1024) gives unacceptable overhead is unknown, but quite likely. If you use heavy raytracing on relatively simple scenes it might work though. Again, if the tiles are square in terms of pixels, then Save Buffers can be used to ease memory.

Limitations & Work−arounds

Blender cannot handle more than 64 parts in the Y direction

Panoramic Rendering

Mode: All Modes

Panel: Render Context Render

Hotkey: F10

Description

To obtain nice panoramic renderings, up to 5 times (!) a full 360° view of the horizon, blender provides an automatic procedure.

Options

"Pano" Button.

You can, by decreasing the focal length of your camera, get a wider field of view, up to 173° (length of 1mm), but at cost of huge distortions in the image ("fish−eye" effect); furthermore, you won't be able to get wider than these 173°.

But Blender is able to render an image showing a 1800° panorama (5 full rotations) of the scene, as if the camera was rotating around its Y axis, with few distortions. For rendering a "real" panorama, enable the Pano button. Henceforth, the behaviour of some render and camera settings are changed:

Camera

Lens:

A 5 (mm) lens setting gives a 360° pano. The horizontal field of view is now proportional to this setting: 10mm gives a 180° pano, 2.5mm gives a 720° pano (two turns), 1mm gives a 1800° pano (5 turns), etc...
***This change only affects the horizontal field of view**: the vertical one behaves as usual (i.e. it is locked to 173° at maximum!). This means that if you want to render a vertical pano, you have to lay the camera on its side.*

Rendering

Xparts

Defines the number of "shots" aligned side by side: at minimum to 10 if you want a "correct" pano; the higher it is, the better is the result (lower distortions); the max number of shots is the width of the rendered picture, in pixels, divided by eight.

Yparts

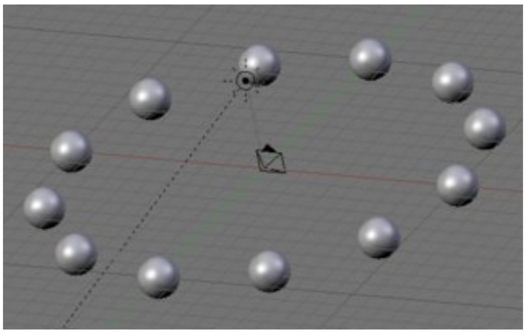
*Its behaviour **isn't changed** from a "standard" rendering.*

SizeX, SizeY

As long as SizeX > SizeY, the horizontal field of view stays the same, as defined by Lens: (e.g., for a 5mm lens, 360°): the vertical field of view is proportional to the ratio height/width.
If SizeX < SizeY, the vertical field of view is locked to its maximum (173° for a Lens: of 1mm, 145° for a Lens: of 5mm, etc.): the horizontal field of view is proportional to the ratio width/height (e.g., for a rendered picture twice as high as wide, and a Lens: of 5, we have an horizontal 180° pano, rather than a 360° one).

Examples

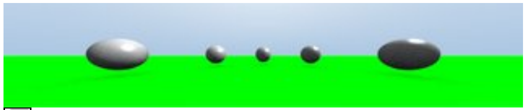
All this is quite complex, so here are some examples, all based on the same scene, to try to clarify it:



Test scene.

Examples of non−panoramic rendering

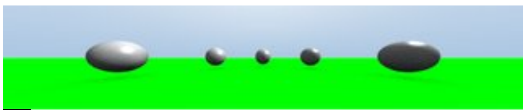
Non−panoramic rendering with Lens: = 1mm (horizontal field of view: 173°).



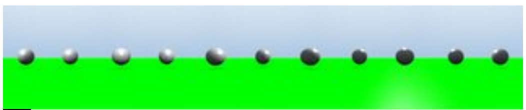
Non−panoramic rendering with Lens: = 5mm (horizontal field of view: 145°).

Non−panoramic rendering with Lens: = 10mm (horizontal field of view: 116°).

Examples of panoramic rendering



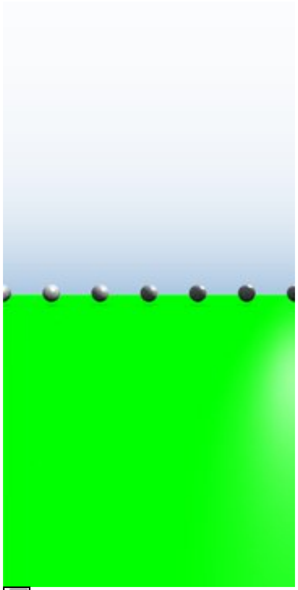
Panoramic rendering with Lens: = 5mm, and Xparts = 1. There's no difference from a rendering with the same lens without the Pano option!



Panoramic rendering with Lens: = 5mm, and Xparts = 5. The distortions are still obvious, and the horizontal field of view is not yet full 360°...

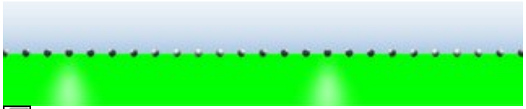
Panoramic rendering with Lens: = 5mm, and Xparts = 10. Nearly perfect.

Panoramic rendering with Lens: = 5mm, and Xparts = 90. Compare with the previous one: very few differences...



Rendu panoramique avec Lens: = 5mm, and Xparts = 90, and twice as high as wide: we have a 180° pano instead of a 360° one, with a vertical field of view of 145 °!

Panoramic rendering with Lens: = 1mm, and Xparts = 90. Five complete turns (very useful!).



Panoramic rendering with Lens: = 2.5mm, and Xparts = 90. Two complete turns (very useful!).

Panoramic rendering with Lens: = 10mm, and Xparts = 90. 180° pano.

Author's note

Everything above about the panoramic rendering is written from my blender user's experience: I've never looked at its renderer code...

Motion Blur Rendering

Mode: All Modes

Panel: Render Context Render

Hotkey: F10

Description

Blender's animations are by default rendered as a sequence of *perfectly still* images. This is unrealistic, since fast moving objects do appear to be 'moving', that is, blurred by their own motion, both in a movie frame and in a photograph from a 'real world camera'. To obtain such a Motion Blur effect, Blender can be made to render the current frame and some more frames, in between the real frames, and merge them all together to obtain an image where fast moving details are 'blurred'.

Options

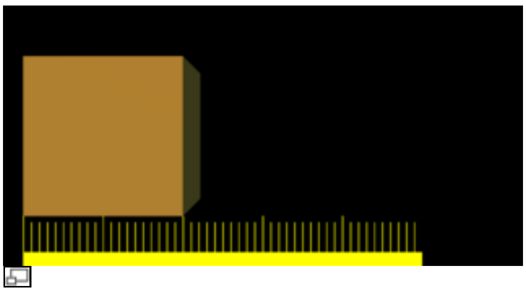
Motion Blur buttons *(F10)*.

MBLUR
: Enables multi−sample motion blur. This makes Blender render as many 'intermediate' frames as the oversampling number is set to (5, 8, 11 or 16) and accumulate them, one over the other, on a single frame. The number−button Bf: or Blur Factor defines the length of the shutter time as will be shown in the example below. Setting the OSA Button is unnecessary since the Motion Blur process adds some antialiasing anyway, but to have a really smooth image OSA can be activated too. This makes each accumulated image have anti−aliasing (becareful with the render time!).

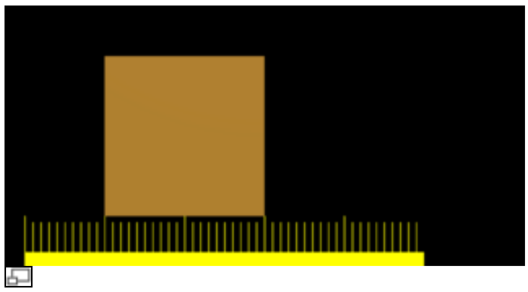
Examples

To better grasp the concept let's assume that we have a cube, uniformly moving 1 Blender unit to the right at each frame. This is indeed fast, especially since the cube itself has a side of only 2 Blender units.

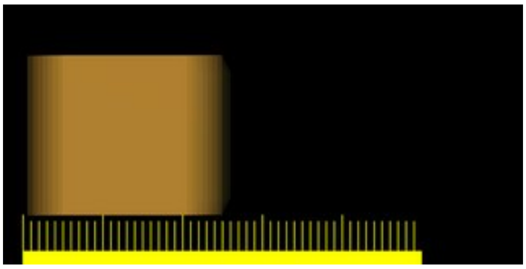
Frame 1 of moving cube without Motion Blur. shows a render of frame 1 without Motion Blur, *Frame 2 of moving cube without Motion Blur.* shows a render of frame 2. The scale beneath the cube helps in appreciating the movement of 1 Blender unit.



Frame 1 of moving cube without Motion Blur.



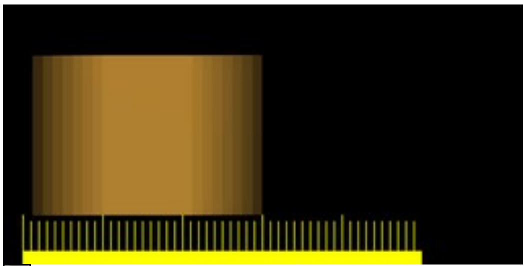
Frame 2 of moving cube without Motion Blur.



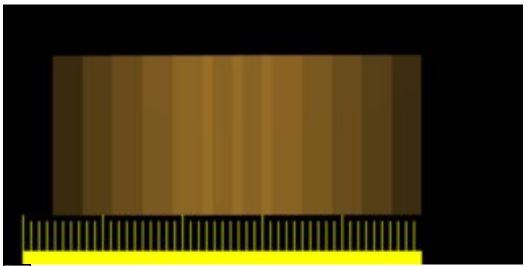
Frame 1 of moving cube with Motion Blur, 8 samples, Bf=0.5.

Frame 1 of moving cube with Motion Blur, 8 samples, Bf=0.5. on the other hand shows the rendering of frame 1 when Motion Blur is set and 8 'intermediate' frames are computed. Bf is set to 0.5; this means that the 8 'intermediate' frames are computed on a 0.5 frame period starting from frame 1. This is very evident since the whole 'blurriness' of the cube occurs on half a unit before and half a unit after the main cube body.

Frame 1 of moving cube with Motion Blur, 8 samples, Bf=1.0. and *Frame 1 of moving cube with Motion Blur, 8 samples, Bf=3.0.* show the effect of increasing Bf values. A value greater than 1 implies a very 'slow' camera shutter.



Frame 1 of moving cube with Motion Blur, 8 samples, Bf=1.0.



Frame 1 of moving cube with Motion Blur, 8 samples, Bf=3.0.

Better results than those shown can be obtained by setting 11 or 16 samples rather than 8, but, of course, since as many *separate* renders as samples are needed a Motion Blur render takes that many times more than a non−Motion Blur one.

Hints

If Motion Blur is active, even if nothing is moving on the scene, Blender actually 'jitters' the camera a little between an 'intermediate' frame and the next. This implies that, even if OSA is off, the resulting images have nice Anti−Aliasing. An MBLUR obtained Anti−Aliasing is comparable to an OSA Anti−Aliasing of the same level, but generally slower.

This is interesting since, for very complex scenes where a level 16 OSA does not give satisfactory results, better results can be obtained using *both* OSA and MBlur. This way you have as many samples per frame as you have 'intermediate' frames, effectively giving oversampling at levels 25,64,121,256 if 5,8,11,16 samples are chosen, respectively.

Stamp

V.: 2.46

Stamps the render with key date/time and other info. See Reference Manual for more info.

Previous: Manual/Oversampling (Antialiasing)	Contents	*Next:* Manual/Render Bake

User Manual: [Contents] | [Guidelines] | Blender Version **2.44**

Baking, in general, is the act of pre−computing something in order to speed up some other process later down the line. Rendering from scratch takes a lot of time depending on the options you choose. Therefore, Blender allows you to "bake" some parts of the render ahead of time, for select objects. Then, when you press Render, the entire scene is rendered much faster, since the colors of those objects do not have to be recomputed.

Render Baking

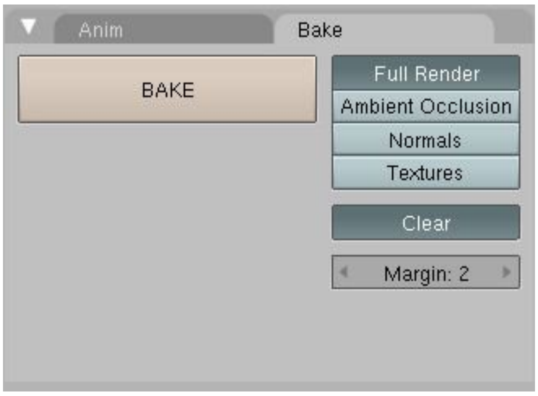
Mode: All Modes except Sculpt

Panel: Scene (F10) Render Context Bake panel

Hotkey: Ctrl Alt B

Menu: Render Bake Render Meshes

Description



The Bake tab in the Render buttons panel.

Render baking creates 2D bitmap images of a mesh object's rendered surface. These images can be re−mapped onto the object using the object's UV coordinates. Baking is done for each individual mesh, and can only be done if that mesh has been UV−unwrapped. While it takes time to set up and perform, it saves render time. If you are rendering a long animation, the time spent baking can be much less than time spent rendering out each frame of a long animation.

Use Render Bake in intensive light/shadow solutions, such as AO or soft shadows from area lights. If you bake AO for the main objects, you will not have to enable it for the full render, saving render time.

Use Full Render or Textures to create an image texture; baked procedural textures can be used as a starting point for further texture painting. Use Normals to make a low−resolution mesh look like a high−resolution mesh. To do that, UV−unwrap a high−resolution, finely sculpted mesh and bake its normals. Save that normal map, and Map To (material settings) the UV of a similarly unwrapped low−resolution mesh. The low−resolution mesh will look just like the high−resolution, but will have much fewer faces/polygons.

Advantages

Can significantly reduce render times
Texture painting made easier
Reduced polygon count
Repeated renders are made faster, multiplying the time savings

Disadvantages
Object must be UV−unwrapped.
If shadows are baked, lights and object cannot move with respect to each other.
Large textures (eg 4096x4096) can be memory intensive, and be just as slow as the rendered solution.
Human (labor) time must be spent unwrapping and baking and saving files and applying the textures to a channel.

You apply the baked Normal image as a Normal Map using a Texture channel.

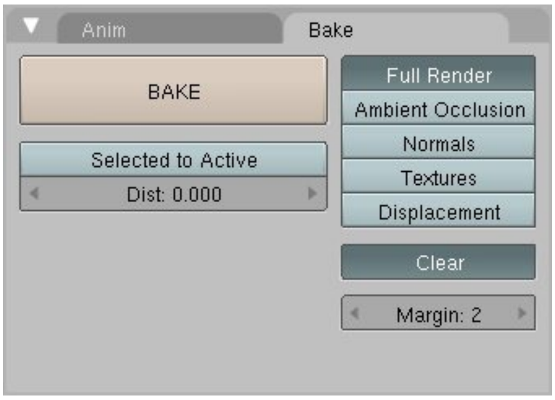
Options

Full Render
Bakes all materials, textures, and lighting except specularity and SSS.
Ambient Occlusion
Bakes ambient occlusion as specified in the World panels (F8). Ignores all lights in the scene.
Normals
Bakes tangent and camera−space normals (amongst many others) to an RGB image.
Textures
Bakes colors of materials and textures only, without shading.
Clear
If selected, clears the image to selected background color (default is black) before baking render.
Margin
Baked result is extended this many pixels beyond the border of each UV "island," to soften seams in the texture.

Mesh Must be Visible in Render

If a mesh not is visible in regular render, for example because it is disabled for rendering in the Outliner or has the DupliVerts setting enabled, it cannot be baked to.

Options under development (after 2.45)



The Bake tab in the Render buttons panel.***Development version***

Read more here: Changes since 2.45 −> Baking Selected to Active

Selected to Active toggle button
: Information from other object can be baked onto the active object, with the Selected to Active option.

Distance
: The Distance parameter controls how far a point on another object can be away from the point on the active object. Only needed for Selected to Active.

  A typical use case is to make a detailed, high poly object, and then bake it's normals onto an object with a low polygon count. The resulting normal map can then be applied to make the low poly object look more detailed.

Displacement Map (new option for render−bake type)
: Similar to baking normal maps, displacement maps can also be baked from a high−res object to an unwrapped low−res object, using the Selected to Active option.

  When using this in conjunction with a subsurf and displacement modifier within Blender, it's necessary to temporarily add a heavy subsurf modifier to the 'low res' model before baking. This means that if you then use a displacement modifier on top of the subsurf, the displacement will be correct, since it's stored as a relative difference to the subsurfed geometry, rather than the original base mesh (which can get distorted significantly by a subsurf). The higher the render level subsurf while baking, the more accurate the displacements will be. This technique may also be useful when saving the displacement map out for use in external renderers.

Normals can now be baked in different spaces:

- Camera space − The already existing method.
- World space − Normals in world coordinates, dependent on object transformation and deformation.
- Object space − Normals in object coordinates, independent of object transformation, but dependent on deformation.
- Tangent space − Normals in tangent space coordinates, independent of object transformation and deformation. This is the new default, and the right choice in most cases, since then the normal map can be used for animated objects too.

For materials the same spaces can be chosen as well, in the image texture options, next to the existing Normal Map setting. For correct results, the setting here should match the setting used for baking. Note that this replaces the NMap TS setting in material options, which is automatically converted to the Tangent space option in the texture options.

Workflow

Windows Users do AO first

If you are running Blender on a Windows operating system, you may have to first bake Ambient Occlusion before baking any other option. If you do not bake AO first, you may get the error message **"No Image to Bake To"** and will not be able to bake anything for that mesh and will have to restart Blender.

1. In a 3D View window, select a mesh and enter UV/Face Select mode
2. Unwrap the mesh object
3. In a UV/Image Editor window, either create a new Image or open an existing Image. If your 3D view is in Textured display mode, you should now see the image mapped to your mesh. Ensure that all faces are selected.
4. With your mouse cursor in 3D View, press Ctrl Alt B to popup the menu of available baking choices. Alternatively, access the Bake panel in the Buttons window, Scene (F10) context, Render sub−context.

5. Bake your desired type of image: Full Render, Ambient Occlusion, Normals, or shadeless Textures.
6. After computation, Blender replaces the image with the Baked image.
7. Save the image in the UV/Image Editor window via Image−>Save
8. Refine the images using the process described below, or embellish with Texture Paint or an external image editor.
9. Apply the image to the mesh as a UV texture. For displacement and normal maps, refer to Bump and Normal Maps. For full and texture bakes, refer to Textures. In all cases you want to Map Input to the UV Map that was used when baking.

Render baking and additional options are available in the Render buttons context, Bake tab. The saved image can now be assigned as an image texture using UV coordinates in Map Input tab of Materials context (F5). In a multiple UV setup, images can be baked from one (inactive) UV set to another (active) set. This can be useful for seam removal, or for game engines that require particular UV setups.

Refining Baked Images

Mode: UV/Image Editor

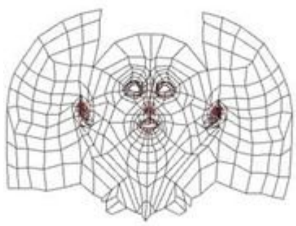
Hotkey: Alt S

Menu: Image−>Save

After baking, be sure to save the generated image. Images may also be refined in the UV/Image Editor using Texture Paint.

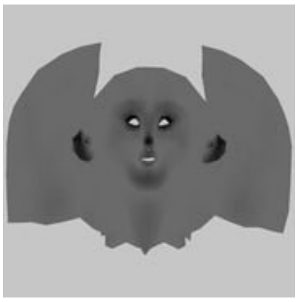
Examples

"Adrianna" model by Mr_Bomb; unwrap and textures by BgDM



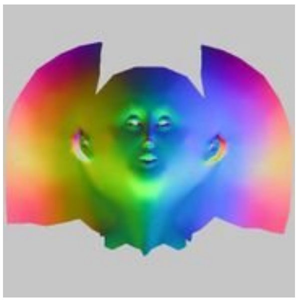
UV Layout for Adrianna model

These examples use a model of a human head, and its UV Layout, shown to the left. The mesh was seamed along the top scalp line and down back of the head, and unwrapped using LSCM projection. It has seven layered image and procedural textures, and is lit by a hemisphere of spots as well as several accent lights.



Ambient occlusion bake

Ambient Occlusion option: Generates a bitmap of the "unwrapped" ambient occlusion result, ignoring all lighting. AO maps are often multiplied with color channels to simulate self−shadowing, soft lighting, or dirt. Note: if the texture shown here were simply multiplied by the diffuse texture, it would darken it by half; see below for a compositing noodle to brighten and blur this image.



Normals bake (camera space)

A Normals bake generates a camera−space normals RGB map. The orientations of the mesh's vertex normals relative to the Camera's X, Y, and Z axes are recorded as different levels of Red, Green, and Blue respectively. In this example, the camera was aimed at the right side of the head model, so the right−side faces are largely blue since they faced the camera more or less straight−on (normals parallel to camera's Z). Camera−space maps have limited utility; this map would allow you to create relief on a lower−poly model, but the illusion would be accurate only if viewed from the right side of the head. The solution to this is a "Tangent space" normal map which is independent of camera position and object deformation. Blender can bake many different types of maps, and to see them, please go to Manual/Bump and Normal Maps.

Textures only bake

The Textures option renders all materials and textures devoid of lighting, AO and shadows (similar to a "shadeless" render). This example baked image comprises the color and bump textures, two Cloud textures, and a subtle linear blend texture. A potential use of such a map might be to multiply it with a baked AO map in the compositor.

Full render bake

Full Render bake takes into account lighting/shadow, materials, vertex colors, bump, and AO if applicable. Full render baking does not include specularity and subsurface scattering, as these are view−dependent parameters. Neither does this mode support halo materials. Full renders can be useful as time−savers for renders of static scenes or objects; for example, soft shadows of a tree or piece of furniture cast by an area light (slow to render) can be baked and "painted" into the scene (very fast render), and will remain convincing as long as the object or light do not move.

Margin option "bleeds" pixels beyond edges of UV map

The Margin option adds the specified number pixels beyond the edge of the UV map. Can be useful for minimizing seams when the image is mapped back onto the mesh. If no value is specified, 1 pixel is added by default.

Applying Baked Images

When you baked the mesh, you were in UV Face Select Mode and the image is created. Save or Pack the image. Then, depending on the Option chosen, you map the rendered texture to the mesh using the following method:

Full Render
: In the object's material settings, enable TexFace, which tells Blender not to compute the material color based on the procedural material settings, but instead use the UV Texture image mapped to the mesh according to the UV Layout.
Alternatively, create a Texture channel for the Material. In the MapInput panel, enable UV and enter the name of the UV Texture ("UVTex" by default). In the MapTo panel, enable "Col" for Color. Activate the Texture buttons subcontext, choose Image as the type of texture, and Load the image by selecting it from the selector list by the button "Load New".
With either mapping method, TexFace or Texture Channel, since the Full Render also takes into account the lighting, you should also enable Shadeless in the Material panel. When rendering the composite scene, Blender does not waste time recomputing the lighting effects on Shadeless materials, saving a lot of render time.

Ambient
: Add a texture channel, and in the MapInput panel, map the texture to UV as described above. In the MapTo panel, enable "Col" but change the mix method to "Multiply". Blender will now multiply the above texture color channels by the AO mask, darkening the colors in the cracks and crevices.

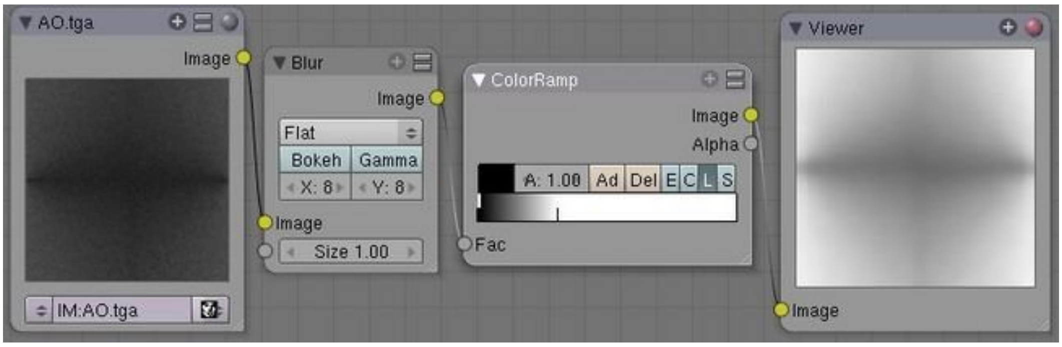
Normal
: Add a texture channel and load the baked normals image. Make sure to activate "Normal Map" in the Map Image panel. In the Materials contents Map Input panel, enable UV; in the Map To panel, enable Nor and turn off "Col". For camera mapped normals, do not use NMap TS in the shader; this option is for tangent space maps only

Texture

The Texture bake includes all materials, but no shading information.

Refining the Ambient Occlusion Image

Often, the rendered AO image will be dark and grainy. Keep in mind this image is multiplied by the base material, so if it is dark, it will darken the whole mesh. If it is grainy, it will make your mesh look speckled or dirty. If so, try increasing the number of *Samples* and disabling *Noise*.



Suggested compositing noodle for AO images

We recommend that you save the AO bake as an image file named "<object>−AO.<ext>", where <object> is the name of the selected object, and <ext> is the appropriate format extension ("tga" for targa format, "jpg" for JPEG, etc). Note that if you have extensions enabled, Blender will add the correct <ext> for you.

As a last resort, although not technically precise, you can blur the image to smooth it out, and increase the dynamic range of the image. To do so, switch to Composite Nodes and construct the noodle shown above. Read in that AO file using the Image input node. The Blur node will remove the graininess, and the adjust the ColorRamp node so that the brightest parts of the image are white (a value of 1.0).

Using a UV/Image Editor window, select "Viewer Node" as the display, and you can then save the image back out; we suggest using something like "<object>−AOK.<ext>". Now, use the AOK file as the multiplier against Color, and it will delicately and smoothly shade the mesh for ambient occlusion.

Applying High−Res Normal to a Low−Res Mesh

One of the tricks of the trade, especially in gaming, is to make a low−poly model look better by modeling two versions of the same thing; one low−poly, and another high−poly. Then, you bake the normal map of the high−poly model, which gives you a texture image that shows all the shadows and creases and detail. Then, apply those images to the low−poly model, and you save yourself a lot of time that you would otherwise have to spend hand−painting the low−poly UV Texture. For detailed explanations on normal maps, please visit Manual/Bump and Normal Maps.

Thanks to the option which allows baking from selected to Active, you do not need to unwrap the high−poly mesh. Here is the steps:

1. Make a copy of your object.
2. Unwrap the copy at a low resolution.
3. Add a new image to the unwrapped mesh using the UV / Image window.

4. Make sure the High−poly and Low−poly objects share the exact same location in Blender (having them on separate layers can help. You can work on one layer at a time and then select both layers when you bake.)
5. Select the High−Poly Object, Shift Select the Low−Poly object. This leaves both selected with the Low−poly being the Active selected object.
6. In the Render panel, select the Bake tab.
7. Choose the map you want to Bake (Normal or Displacement).
8. Choose the "Selected to Active" button.
9. Click Bake.

Hints

Currently (v2.45), only mesh objects can be render−baked.

Baked images are not automatically saved. Save from the UV editor window with Alt−S or Image menu −> Save.

Baking can be interrupted with the Esc key.

If you move lights, you should re−bake any Full Render textures. If you change the mesh geometry, you should re−bake all of the options.

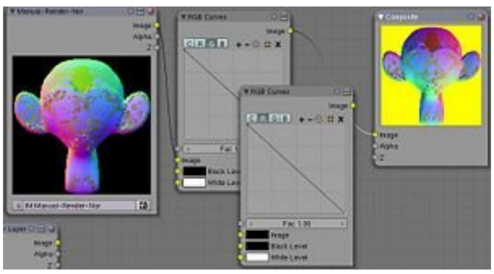
If you change any object's material color or any settings, you have to re−bake that object's Texture or Full Render for those changes to take effect.

Procedural textures (e.g. Clouds) can be baked with the Full Render or Textures options, providing a basis for further detailing. Use Shadeless materials if necessary.

If you get the error "No images found to bake to", check the following:

1. The object is selected
2. Ensure there is a new image (with UV test grid) loaded for baking.
3. If Step 1 does not work although object meets all criteria in that Step, switch to Object mode, select the object, and try to bake.
4. Try Image−>New again in the UV editor and bake to AO before anything else.

Blender to xNormal/Maya Normal



Converting Blender Normal to xNormal

Blender's normal map baking tool is different from other applications, such as xNormal and Maya. Both of these are widely used in the games industry and are already production proven (meaning the way they bake

their normal maps is more commonly used and can be regarded as a standard). To convert a Blender normal map to an xNormal, you need to invert the Red and Green channels. This is easily accomplished using the noodle shown to the right. Use the Image input node to read in the baked normal from Blender. Thread the Image socket to an RGB Curves node. In the Red and Green channels, drag the left endpoint up, and the right endpoint down, so that the curve runs diagonally down as shown. Thread the Image output to the Composite output, enable Do Composite, and save the image.

Previous: Manual/Rendering Options	Contents	*Next:* Manual/Rendering Animations

User Manual: Contents | Guidelines | Blender Version **2.44**

Rendering Animations

Mode: All Modes

Panel: Render Context Anim

Hotkey: F10

Description

While rendering stills will allow you to view and save the image from the render buffer when it's complete, animations are a series of images, or frames, and are automatically saved directly out to disk after being rendered.

When using Blender, the idea is to use the Compositor to do green screen masking, matting, color correction, DOF, and so on to the images. That result is then fed to the Sequencer where the strips are cut and mixed and a final overlay is done.

Workflow

Generally, you do a lot of intermediate renders of different frames in your animation to check for timing, lighting, placement, materials, and so on. At some point, you are ready to make a final render of the complete animation for publication.

There are two approaches you can use when making a movie, or animation, with or without sound. The approach you should use depends on the amount of CPU time you will need to render the movie. You can render a "typical" frame at the desired resolution, and then multiply by the number of frames that will ultimately go into the movie, to arrive at an total render time. If the total render time is an hour or more, you want to use the "Frame Sequence" approach. For example, if you are rendering a one−minute video clip for film, there will be (60 seconds per minute) * (24 frames per second) or 1440 frames per minute. If each frame takes 30 seconds to render, then you will be able to render two frames per minute, or need 720 minutes (12 hours) of render time.

Rendering takes all available CPU time; you should render overnight, when the computer is not needed, or set to a low priority while rendering, and work on other things (be careful with the RAM space!).

The **Direct Approach** is where your set your output format to an AVI or MOV format, and click ANIM to render your scene directly out to a movie file. Blender creates one file that holds all the frames of your animation. You can then use Blender's VSE to add an audio track to the animation and render out to an FFMPEG format to complete your movie.

The **Frame Sequence** approach is where you set your output format to a still format (such as JPG, PNG or MultiLayer), and click ANIM to render your scene out to a set of images, where each image is the frame in the sequence. Blender creates a file for each frame of the animation. You can then use Blender's compositor to perform any frame manipulation (post processing). You can then use Blender's VSE to load that final image sequence, add an audio track to the animation, and render out to an FFMPEG format to complete your movie. The Frame Sequence approach is a little more complicated and takes more disk space, but gives you more flexibility.

Here are some guidelines to help you choose an approach.

Direct Approach

short segments with total render time < 1 hour
stable power supply
computer not needed for other uses

Frame Sequence Approach

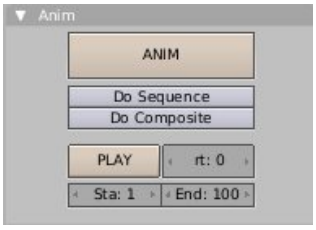
total render time > 1 hour
post−production work needed
Color/lighting adjustment
Green screen / matte replacement
Layering/compositing
Multiple formats and sizes of ultimate product
intermediate frames/adjustments needed for compression/codec
precise timing (e.g. lip−sync to audio track) needed in parts
may need to interrupt rendering to use the computer, and want to be able to resume rendering where you left off.

Frame Sequence Workflow

1. . First prepare your animation.
2. . In the Scene panels setup your animation to be rendered out as a image, generally using a format that does not compromise any quality (I prefer PNG or MultiLayer because of their lossless nature).
3. . Choose the output path and filespec in the Output render panel, for example "//\render\my−anim−"
4. . Confirm the range of your animation frame Start and End
5. . Save your .blend file.
6. . Press the big *ANIM* (animate) button. Sleep or get a life while you wait for your computer to do its thing.
7. . Once the animation is finished, use your OS file explorer to navigate into the output folder (".\render in this example). You will see lots of images (.png or .exr, etc... depending on the format you chose to render) that have a sequence number attached to them ranging from 0000 to a max of 9999. These are your single frames.
8. . In Blender, now go into the video sequence editor.
9. . Choose *Add Image* from the add menu. Select all the frames from your output folder that you want to include in your animation. They will be added as a strip to the sequence editor.
10. . Now you can edit the strip and add effects or simply leave it like they are. You can add other strips, like an audio strip.
11. . Scrub through the animation, checking that you have included all the frames.
12. . In the Scene Render buttons, in the Anim panel, below the animation button, activate *Do Sequence*.
13. . In the Format panel, choose the container and codec you want (e.g. FFMPEG H.264) and configure it.
14. . Click the ANIM button and Blender will render out the sequence editor output into your movie.

Why go through all this hassle you may ask? Well, first of all, if you render out single frames you can stop the render at any time by pressing ESC in the render window. You will not lose the frames you have already rendered, since they have been written out to individual files. You can always adjust the range you want to continue from where you left off. You can edit the frames after wards and postprocess them. You can add neat effects in the sequence editor. You can render the same sequence into different resolutions (640x480, 320x240, etc) and use different codecs (to get different file sizes and quality) with almost no effort what's so ever.

Options



Animation rendering buttons.

ANIM
: Starts the animation rendering.

Do Sequence
: Renders the output of the sequence editor, instead of the view from the 3D scene's active camera. If the sequence contains scene strips, these will also be rendered as part of the pipeline. If Do Composite is also enabled, the Scene strip will be the output of the Compositor.

Do Composite
: Renders the output from the Compositing noodle, and then pumps all images through the Composite node map, displaying the image fed to the Composite Output node.

Play
: Opens a modal render window and plays the animation for you. You must close this window before returning to Blender via Esc or using the window control X. A nice feature of this window is that the frame number is shown in the header. If you LMB click in the window, the video will reset. Holding down your up−arrow will move the video forward in real−time; holding the down arrow reverses the video, thus allowing you to scrub through the video to identify exactly any errant frames.

Carried Away

If you render out to a frame sequence of 100 frames for example, and then change your Animation Sta: and End: to something smaller, say 1 to 30, Blender will play all the frames it can find in that sequence (all 100). If you re−render with changes to update that smaller subset, Blender will still play the new set plus the rest of the old set. The only solution is to modify the output file name before re−rendering!

Return Code
: Useful for debugging

Sta and End
: The start and end frame numbers to render the animation from and to. Frame numbers are inclusive. These are also set if you modify the Sta: and End: fields in the Timeline window.

Animation location and extensions.

By default the animation is rendered in the directory specified in the Output Panel (*Animation location and extensions.*). If an AVI format has been selected, then the name will be ####_####.avi where the '####' indicates the start and end frame of the animation, as 4 digit integers padded with zeros as necessary.

If an image format is chosen, on the other hand, a series of images named ####, ('####' being the pertinent frame number) is created in the directory.

Extensions
: Adds the correct file extensions per file type to the output files

Hints

Argh! My bratty sister turned off the PC right in the middle of rendering my movie!
: Unless your animation is really simple, and you expect it to render in half an hour or less, it is always a good idea to render the animation as separate image frames in a lossless format (TGA, PNG, BMP) rather than as a movie file from the beginning. This allows you an easy recovery if there is a problem fails and you have to re−start the rendering, since the frames you have already rendered will still be in the Output directory. Just change the STArt frame number to the frame number where you want to pick up from, and click ANIM again.

I only need to re−render a few frames in the middle
: It's also a good idea to render initially to a frame sequence, since if only a few frames have an error, you can make corrections and re−render just the affected frames. You can then make a movie out of the separate frames with Blender's sequence editor or with compositing nodes.

Only first frame renders, then Blender locks up
: If you click ANIM and only the first frame renders, be sure the output file is not locked by the media player. In general, check the console when rendering.

Unable to create Quicktime movie
: CreateMovieFile error: −47
The Quicktime movie strip is in use (possibly in the VSE) and cannot be overwritten. If it is used in the VSE, delete the strip, or delete the file using your file explorer.

Previous: Manual/Render Bake	Contents	*Next:* Manual/Output Formats

User Manual: Contents | Guidelines | Blender Version **2.43**

Output Formats

Mode: All Modes

Panel: Render Context Format

Hotkey: F10

Description

The output format for either single image renders (**RENDER** F12 + F3 save image) or for Animations **ANIM** Ctrl F12 is selected using the Format Panel. From here you can select many image or animation formats (*Image and animations formats.*).

There are many image formats out there for many different uses. A format stores an image in a *lossless* or lossy format; with lossy formats you suffer some image degredation but save disk space because the image is saved using fewer bytes. A lossless format preserves the image exactly, pixel for pixel. You can break formats down into *static* images and movie *clips*. Within either category there are standards (static formats and clip codecs) which may be proprietary standards (developed and controlled by one company), or open standards (which are community or consortium−controlled). Open standards generally outlive any one particular company and will always be royalty−free and freely obtained by the viewer. Proprietary formats may only work with a specific video card, or while the codec may be free, the viewer may cost.

Format Panel

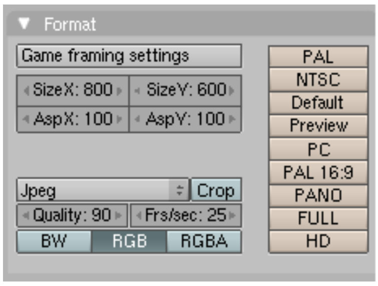


Image Format options.

By default the dimensions SizeX and SizeY are 800Ã—600 and can be changed as for any Num Button by shift−clicking into the field. These buttons control the overall size of the image. Just below are two more settings, AspX and AspY which control the packing of the pixels along the respective axis. Together, the four buttons below define the viewing size and aspect ratio of the image. Aspect is the ratio between the X and Y dimensions of the pixel of the image. By default it is 1:1 since computer screen pixels are square. If television shorts are being made, and since TV pixels are not square, you want to change this aspect ratio to match the destination video standard: PAL for Europe, and NTSC for the Americas.

Presets

To make life easier the rightmost block of buttons provide some common presets:

- PAL − 720Ã—576 pixels at 54:51 aspect ratio.
- NTSC − 720Ã—480 pixels at 10:11 aspect ratio.
- Default − Same as PAL, but with full TV options, as explained in the following sections.
- Preview − 640Ã—512 at 1:1 aspect ratio. This setting automatically scales down the image by 50%, to effectively produce a 320Ã—256 image.
- PC − 640Ã—480 at 1:1 aspect ratio.
- PAL 16:9 − 720Ã—576 at 64:45 aspect ratio, for 16:9 wide screen TV renderings.
- PANO − Standard panoramic settings 576Ã—176 at 115:100 aspect ratio. More about 'panoramic' renderings in the pertinent section.
- FULL − 1280Ã—1024 at 1:1 aspect ratio.
- HD − Maximus Pixelus. With a 16:9 aspect ratio and checking in at a full 1920Ã—1080 resolution for ever frame, this is the ultimate format, with over 2 million pixels to compute for every frame. Bassam says to "Grab a cup of coffee while you wait, but the results are spectacular." You can also set format to 1280x720 30 fps to get HD 720p flavor. Check the Even Interlacing button for the "i" variety of 1080/720, thus giving support for all HighDef varieties.

These are just the presets; you can set any resolution you wish, subject to your PC's memory restrictions; see the Render page for ideas and techniques and tools for enabling huge render outputs.

Options

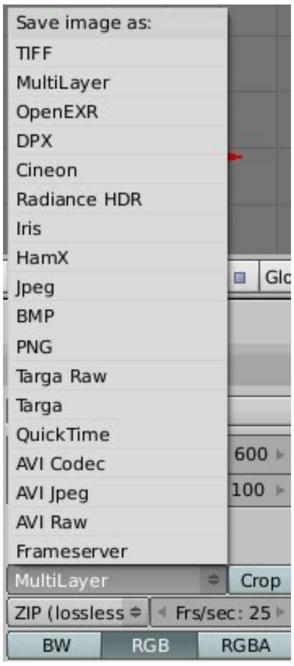


Image and animations formats.

Blender supports a wide mix of image formats. Some formats are produced by internal Blender code. Others (Tiff, for example) may require a dynamic load library (such as libtiff.dll) in your Blender installation folder. In alphabetical order they are (**bold** indicates a movie clip format):

- **AVI Raw** − Audio−Video Interlaced (AVI) uncompressed frames.
- **AVI Jpeg** − AVI but with Jpeg compression. Lossy, smaller files but not as small as you can get with

a Codec compression algorithm. Jpeg compression is also the one used in the DV format used in the digitals camcorders.

- **AVI Codec** − AVI codec compression. Available codecs are operating system dependant. When an AVI codec is initially chosen, the codec dialog is automatically launched. The codec can be changed directly using the Set Codec button which appears (*AVI Codec settings.*).
- BMP Bit−Mapped Paint lossless format used by early paint programs.
- Cineon − format produced by a Kodak Cineon camera and used in high−end graphics software and more directed toward digitial film.
- DPX − Digital Moving−Picture eXchange format; an open professional format (close to Cineon) that also contains metainformation about the picture; 16−bit uncompressed bitmap (huge file size). Used in preservation.
- Frameserver − Blender puts out frames upon request as part of a render farm. The port number is specified in the OpenGL User Preferences panel.
- HamX − Blender's own self−developed 8 bits RLE (Run Length Encoded bitmap) format; it creates extremely compact files that can be displayed quickly. To be used only for previsualization of animations (Play button).
- Iris − the standard Silicon Graphics Inc (SGI) format used on those spanking Unix OS machines.
- Jpeg − Joint Picture Expert Group (name of the consortium which defined it), an open format that supports very good compression with little loss of quality. Only saves RGB value. Re−saving images results in more and more compression and loss of quality.
- MultiLayer − an OpenEXR format that supports storing multiple layers of images together in one file. Each layer stores a renderpass, such as shadow, specularity, color, etc. You can specify the encoding used to save the MulitLayer file using the codec selector (ZIP (lossless) is shown and used by default).
- OpenEXR − an open and non−proprietary extended and highly dynamic range (HDR) image format, saving both Alpha and Z−depth buffer information.

    ◊ Enable the *Half* button to use 16−bit format; otherwise 32−bit floating point precision color depth will be used
    ◊ Enable the *Zbuf* button to save the Z−buffer (distance from camera) info
    ◊ Choose a compression/decompression *CODEC* (ZIP by default) to save disk space.
    ◊ Enable the *RGBA* button to save the Alpha channel.
    ◊ Because OpenEXR is so new and previews are generally not supported by Operating Systems, enable *Preview* to save a JPG image along with the EXR image so you can quickly and easily see what the basic image looks like.

- PNG − Portable Network Graphics, a standard meant to replace old GIF inasmuch as it is lossless, but supports full true colour images. Supports Alpha channel.

  Enable the RGBA button to save the Alpha channel.

- Radiance HDR − a High Dynamic Range image format that can store images with big changes in lighting and color.
- TARGA and Targa raw − Truevision Advanced Raster Graphics Adapter is a simple raster graphics format established in 1984 and used by the original IBM PC's. Supports Alpha Channel.

  Enable the RGBA button to save the Alpha channel.

- TIFF − Often used for teletype and facsimile (FAX) images
- **QuickTime** − Apple's Quicktime .mov file. The Quicktime codec dialog is displayed when this codec is installed and this format is initially chosen.

Reads GIF if QuickTime is Installed

Blender can read GIF files on Windows and Mac platforms with [QuickTime] installed. The GIF capabilities (as well as flattened PSD, flattened PDF on Mac, and others) come along with QuickTime.

Compression

Some formats can compress the image to use less disk space. This compression might be lossless (PNG, ...) or lossy (Jpeg, ...). Lossy formats don't store individual pixel information, thus reducing image quality. All the other formats are more or less equivalent, each having advantages and disadvantages. Make your compression selection using the button or field located beneath the format selector. For example, if Jpeg is selected, you can specify a compression level (Quality:90 by default). Higher quality takes more disk space, but results in a better looking picture with less compression encoding artifacts.

The default image type is Targa, but, since the image is stored in a buffer and then saved, it is possible to change the image file type after the rendering and before saving using this menu. (**attention**: this is only valid for static images, not when rendering animations!).

Channels

Blender renders color (RGB) images, but Black and White (BW) and color with Alpha Channel (RGBA) are also possible. Beware, unless the Extensions button of the Output pannel is set, Blender does *not* automatically add the extension to files, hence any .tga or .png extension must be explicitly written in the File Save window.

If the option "RGBA" is specified, standard colour number '0' is used as the transparent colour. The format MUST support an alpha channel as part of its specifcation; for example, if you choose a Jpeg format (whose specifcation does not support alpha transparency), clicking RGBA will not magically add an alpha channel; it will be saved as RGB only.

OpenEXR and **Multilayer** formats are the only formats that store Z−depth buffer information. **Multilayer** is the only format that stores Render Layer and Render Passes as channels that can then be composited in post−production.

Blender reads and writes (Amiga) IFF, Targa, (SGI) Iris and CDinteractive (CDi) RLE colormap formats. Specify the colormap file in the Output FType field.

Frame Rate

For an animation the frame rate (*AVI Codec settings.*) which, by default, is 25 frames per second, the standard for PAL (European) television. Use 30 frames per second for USA television.

Codec

AVI Codec settings.

A Codec is a little routine that compresses the video so that it will fit on a DVD, or be able to be streamed out over the internet, over a cable, or just be a reasonable file size. Codecs compress the channels of a video down

to save space and enable continuous playback. *Lossy* codecs make smaller files at the expense of image quality. Some codecs, like H.264, are great for larger images. Codecs are used to encode and decode the movie, and so must be present on both the encoding machine (Blender) and the target machine. The results of the encoding are store in a container file.

Blender knows two kinds of container files:

- Audio Video Interlace (a .avi extension) and
- QuickTime (a .mov extension).

When AVI Codec is selected, Blender will popup a little Codec selector window, listing the codecs that are registered on your machine. Each Codec has unique configuration settings. Consult the documentation on the codec (supplied by the company that wrote it) for more information.

When Quicktime is selected, the codecs on your machine will pop−up and allow you to pick which one you want to use. You may have to have purchased Quicktime Pro to use this.

There are dozens, if not hundreds, of codecs, including XviD, H.264, DivX, Microsoft, and so on. Each has advantages and disadvantages and compatibiliity with different players on different operating systems.

Most codecs can only compress the RGB or YUV color space, but some support the Alpha channel as well. Codecs that support RGBA include:

- animation (quicktime)
- PNG *TIFF *Pixlet − not lossless, and may be only available on Apple Mac.
- [Lagarith Lossless Video Codec]

More information on image formats can be found at:

- http://www.cineon.com/
- http://www.digitalpreservation.gov/

Interlacing

Interlacing is a way of providing a sort of motion blur with compression. Instead of capturing the full resolution image every so many times a second, half of the horizontal scan lines are captured twice the number of times a second. So, instead of displaying 1280x720 images 25 times a second, which is called HD 720p EU, you could display 1280x360 images 50 times a second, where the first frame is the even scan lines (horizontal rows 2, 4, 6, 8 ...) and the second frame is set 1/50th of a second later consisting of the odd scan lines (rows 1, 3, 5, 7,...). The net result is that the same number of pixels are displayed every second, but the interlaced variety will appear smoother, since the odd lines catch any movement that happened in between the even frames, and vice versa. Blender supports Even interlacing (described above, used for EU TV) and Odd interlacing, for US TV, where the first frame is scan lines (1, 3, 5, 7 ...) and the frame after is the even lines (2, 4, 6, 8, ...). Use the Even/Odd buttons for this purpose.

Frame Rate

Additionally, Blender supports 50 or 60, or 24 or 30 frames per second. 50 and 25 fps are used for EU TV, and 60 and 30 are US, and 24 is film. Because of US Power Grid cycles, the actual frame rate is 29.97 fps. To accommodate this, Blender has a divider for the frame rate field; enter 30 fps and a divider of 1.001 to get

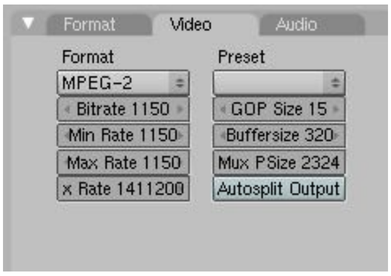
exactly 29.97 fps.

FFMPEG

FFMPEG, short for Fast Forward Moving Pictures Expert Group, is a collection of free and open source software libraries that can record, convert and stream digital audio and video in numerous formats. It includes libavcodec, an audio/video codec library used by several other projects, and libavformat, an audio/video container mux and demux library.

When you select FFMPEG as your output format, two more tabs appear that allow you to select the video codec and the audio codec.

FFMPEG Video



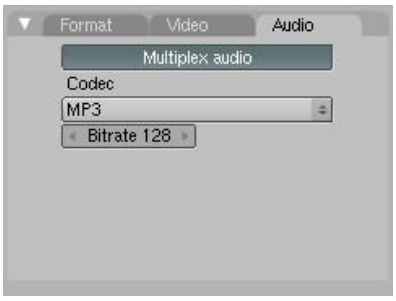
Here you choose which video codec you want to use, and compression settings. With all of these compression choices, there is a tradeoff between filesize, compatibility across platforms, and playback quality. You can use the presets, DV, SVCD, DVD, etc. which choose optimum settings for you for that type of output, or you can manually select the format (MPEG−1, MPEG−2, MPEG−4, AVI, Quicktime (if installed), DV, H.264, or Xvid (if installed). You must have the proper codec installed on your computer for Blender to be able to call it and use it to compress the video stream.

If your video is HUGE and exceeds 2Gig, enable Autospilt Output. The main control over output filesize is the GOP, or keyframe interlace. A higher number generally leads to a smaller file, but needs a higher−powered device to replay it.

Standards

Codecs cannot encode off−the−wall video sizes, so stick to the XY sizes used in the presets for standard TV sizes.

FFMPEG Audio



Audio is encoded using the codec you choose, as long as you enable Multiplex Audio. For each codec, you may be able to control the bitrate (quality) of the sound in the movie. This example shows MP3 encoding at 128kbps. Higher bitrates are bigger files that stream worse but sound better. Stick to powers of 2 for compatibility.

Free Advice

Choosing which format to use depends on what you are going to do with the image. If you are going to

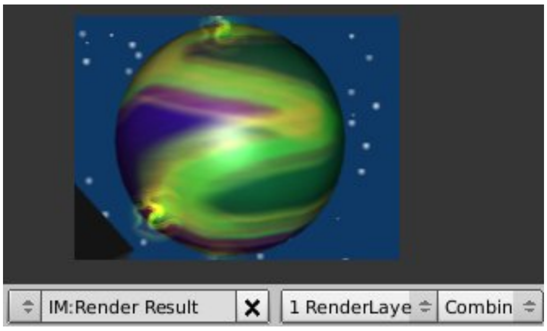
- email it to your friends, use **JPG**
- combine it with other images in post processing and simple color/alpha composition, use **PNG**
- use nodes to simulate depth of field and blurring, use **EXR**
- composite using Render Passes, such as the Vector pass, use **Multilayer**.

If you are animating a movie and are not going to do any post−processing or special effects on it, use either **AVI−JPEG** or **AVI Codec** and choose the XviD open codec. If you want to output your movie with sound that you have loaded into the VSE, use **FFMPEG**.

If you are going to do post−processing on your movie, it is best to use a frame set rendered as **PNG** images; if you only want one file, then choose **AVI Raw**. While AVI Raw is huge, it preserves the exact quality of output for the post−processing. After post−processing (compositing and/or sequencing), you can compress it down. You don't want to post−process a compressed file, because the compression artifacts might throw off what you are trying to accomplish with the post−processing.

Note that rendering an animation long to calculate in a unique file (AVI or QuickTime) is more risky than in a set of static images: if a problem occurs while rendering, you have to re−render all from the begining, while with static images, you can restart the rendering from the place (the frame) where occured the problem!

Checking Render Results



When Blender renders an image, it saves it in an internal buffer. To see the latest render, change one of your panes to a UV/Image Editor. In the IM: Image selector, select Render Result, and the window will display the latest render. To the right of the Image selector, you can also select a RenderLayer to view, and, if you have render passes enabled, the results of a specific pass.

Previous: Manual/Rendering Animations Contents *Next:* Manual/Render Layers

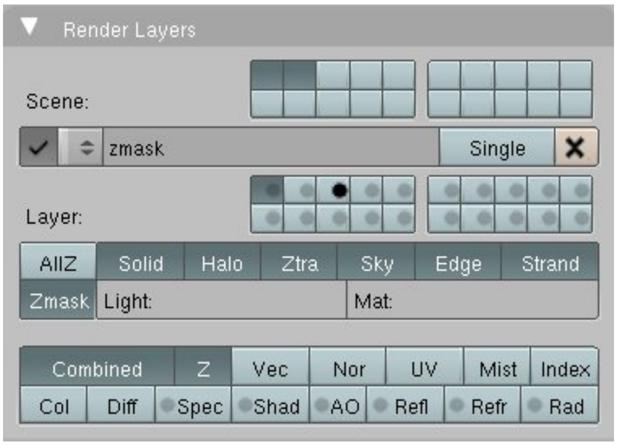
User Manual: Contents | Guidelines | Blender Version **2.46**

Renderlayers are used to separate your composite image into layers. Use RenderLayers for a specific reason − such as creating depth of field, relighting isolated elements within the image via a normal pass, adding a colorcast to specific portions of the image, etc. The keyword here is isolation. Renderlayers allow you to dissect, effect and or correct individual elements or groups within your composition before outputting your final render. This saves you from endlessly re−rendering your scene just to find out whether a correction is going to work or not.

This page is being updated IN ANTICIPATION of some of the changes expected in the NEXT release of Blender.

Specific featurees and options discussed may not be available in the current release.

Render Layers



Render Layer buttons.

This panel allows you control over which layers to render, separate from which layers are presently displayed in your 3−D window. In general, you want objects on all layers to be test−rendered while you work on tweaking objects on only one layer, and hence, display only one or two layers in your workspace.

For example, some of us keep the camera and lights on layer 20, but do not want them cluttering up our workspace. When we render, however, we always want to include that layer, so we select it here.

The dynamic list of RenderLayers and the corresponding Layer buttons allow you to create sets of layer selections. By default, "1 RenderLayer" is all layers. If you click the up−down button and select "ADD NEW", the field changes to "2 RenderLayer". You now select which layers should participate in the "2" set by clicking the appropriate layer buttons (shift−click to select multiple).

For example, imagine a complicated scene having background props, people, cars, buildings, main props, airplanes, spaceships, and tidal waves. Each of these would be kept on a separate layer. As you do test renders while tweaking the poses of the people, you don't want to waste all your CPU power and time rendering background props. So, you create a RenderLayer set that only has the people and lights and cameras (and maybe the spaceship that they are being vaporized by).

The Sky, Solid, Halo, etc. buttons allow you to toggle rendering of the sky, faces of objects with solid

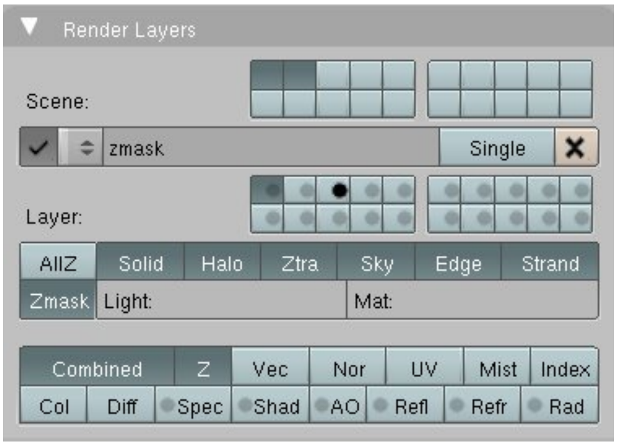
materials (procedural materials only), Halo materials, etc., further allowing you to focus your precious CPU render time of complicated scenes.

You can make the render engine skip certain Passes in order to get an image rendered faster but by sacrificing some feature, for example, Z transparency.

Speed

The Blender Internal rendering engine was built for speed. Most of the time, if you have a half−decent computer and a reasonably complicated scene, you won't need to scale down the render (via all these selections) in order to obtain a pretty fast render.

Finding Render Layers



In the Buttons Window, specifically the Scene Render buttons, the first panel has two tabs: Output and RenderLayers. You're close, if not already there. Click the RenderLayers tab header. This is where you select the layers that you want to render, and the settings for the upcoming render.

By default, the active layers in your scene are also active here. You know this because you have acutely observed that there are TWO sets of layer buttons here: one above the Render Layer name, and another set below (that has all the buttons selected). However, you can change that set of layers here; simply (Shift LMB to select multiple layers that you want to render, or LMB to select a single layer to render.

Layer Sets AND each other

Only the objects in layers that are selected BOTH in the main Scene Layer group AND the Render Layer Layer group will be rendered. So, if the Scene has only Layer 1 selected, and your Render Layer set specifies to render only Layers 2 and 3, nothing but the Sky (if selected) will be rendered.

Using Render Layers

To use RenderLayers, well, you probably already have. When creating your scene, you probably (hopefully) put different objects on different layers.

The real power of Render Layers comes compositing using Blender's node−based compositing system. In Nodes, when you added an Input Node of type RenderLayers, and selected the Scene, you brought in the current renderlayer. That's how you use a RenderLayer: add a Renderlayer input node to your map, and plug it

in.

Render Layer settings

Render Layers are set up via four main groups of options:

- Enabling and Naming
- Scene Layers to Render
- Pipeline Products to deliver
- Passes to make available

Enabling and Naming

The checkbox enables or disables the computation of the whole render layer. Enable only those layers you are working with to save time. The selector allows you to scroll through and examine existing render layers, or to add a new one. You may name your render layers by Shift LMB into the name field. The "Single" button makes Blender render ONLY that render layer and none other. Enable this if you are working on a single layer to avoid rendering unwanted layers. The X deletes the layer.

Scene Layers to Render

The next group of controls are the layers to render. As noted above, this set of layer buttons are ANDed with the currently selected scene layers (shown at the top of the panel). If both corresponding buttons are enabled, the layer is fed into the render pipeline.

You can select that layer totally by LMB clicking the button. To select multiple layers, Shift LMB click

You can select that layer to feed the Z mask (explained below) via Ctrl LMB , and the button's dot turns dark black. Deselect by Ctrl LMB again.

Pipeline Products to deliver

Each renderlayer has its own set of major products to include in the rendering pipeline. To save time and give you control when working with passes, this set of buttons allow you to select which major products to render:

AllZ
: Z−values are computed for everything in view, not just those things that are rendered. When disabled, objects not included in the render have no (infinite) z value

Solid
: Solid faces are rendered. All normal meshes are solid faced.

Halo
: Halo materials are rendered

Z−transparency
: Transparency may be Z−based or Ray−traced. If Z−based, enabling *Ztra* renders transparent areas with the z−value of what is behind the transparent area.

Sky
: Turning on Sky renders the sky, as defined in your material world settings. Otherwise, a black alpha transparent background is rendered

Edge
: If Edge is enable in the Output panel, objects in this renderlayer are given an outline edge. Turning on Edge pulls in the Edge settings from the Output tab, and adds an outline to the objects. Edges have to also be enabled on the Output tab.

Strand
: Strands are strings of static particles that are colored as part of the material settings; they look like strands of hair, or grass.

Z−mask
: The image rendered is from the objects that are between the selected layer(s) and the Z−mask layers. In the example, the cube is on layers 2 and 3, and the grass in on layer 1. In the render layer called "zmask", layer 1 is selected and Z−mask layer 3 is designated. Therefore, only that part of Layer 1 which is in front of the object on layer 3 (the cube) is rendered

Light
: Enter the name of a light group, and the scene will be lit with only those lights. Usually, you use this to speed up draft renders, of a scene that has complicated lighting, by entering the name of a small group of key lights.

Mat
: Overrides all material settings to use the name of the Material entered. Use this to speed up draft renders. Use the default material to check basic lighting.

Passes to make available

Render Passes (Combined, Z, Vec, etc.) are discussed on the next page.

Creating a new Render Layer

By default, there is 1−Renderlayer created for you, and it includes all layers, whether they are used in your scene or not. To add yet another Render Layer, click the yellow up−down selector and select Add New render layer. You now have two Render Layers to choose from, and the active one is shown in the window. Each Render Layer will have its own set of layers that are rendered (sort of makes sense now, doesn't it?).

For example, you might have a robot in a scene with a ground object, buildings, etc. If the robot is on visible−layer 5, you would create one render layer named "Robot" with layer 5 selected in both the Scene: and Layer: buttons.

You would create another render layer (maybe named "stuff") that had all other layers EXCEPT layer 5 selected in both the Scene: and Layer: buttons. Then, back in the Node Editor, you would create TWO input nodes of type Render Layer: one for the Robot Render Layer, and another for the other Stuff. Run both through a mixer and out to the Composite viewer to get the big picture.

Examples

Rendering only certain objects

For example, suppose you have added a cool halo to your robot and you want to quickly see what it looks like. Suppose your scene has boxes on layer 1, laser rifles on layer 2, the robot on layer 5, and lights and camera on layer 20, and they are all selected and visible in 3d view. If you want to render just your robot, and he is on layer 5, you click on the render layer 5 button (which is below the Render Layer name), de−select sky (so that

the sky/horizon is not rendered) and select Halo. Presto! When you render, only the robot is rendered (quickly) and not all the other elements of your scene (like the boxes he is running in front of).

Outlining only selected objects

To render an image where only one or two of the objects are outlined, move those objects onto layer(s) separate from everything else. Create a 1−Render Layer for those layer(s) by selecting only those layers in the Render Layer layer set. Create another 2−Render Layer for the other stuff. Enable the Edge option for 1−Render Layer (remember to also enable Edge on the Output tab) and make sure it is de−selected (off) for 2−Render Layer. In the Node Editor, create two input nodes, one for each Render Layer. Mix the two images. Done. Simple. Yea.

Previous: Manual/Output Formats	Contents	*Next:* Manual/Render Passes

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.43**

Render Passes are the different things the Blender Render Engine must calculate to give you the final image. In each 'pass' the engine calculates different interactions between objects

RenderPass In Detail

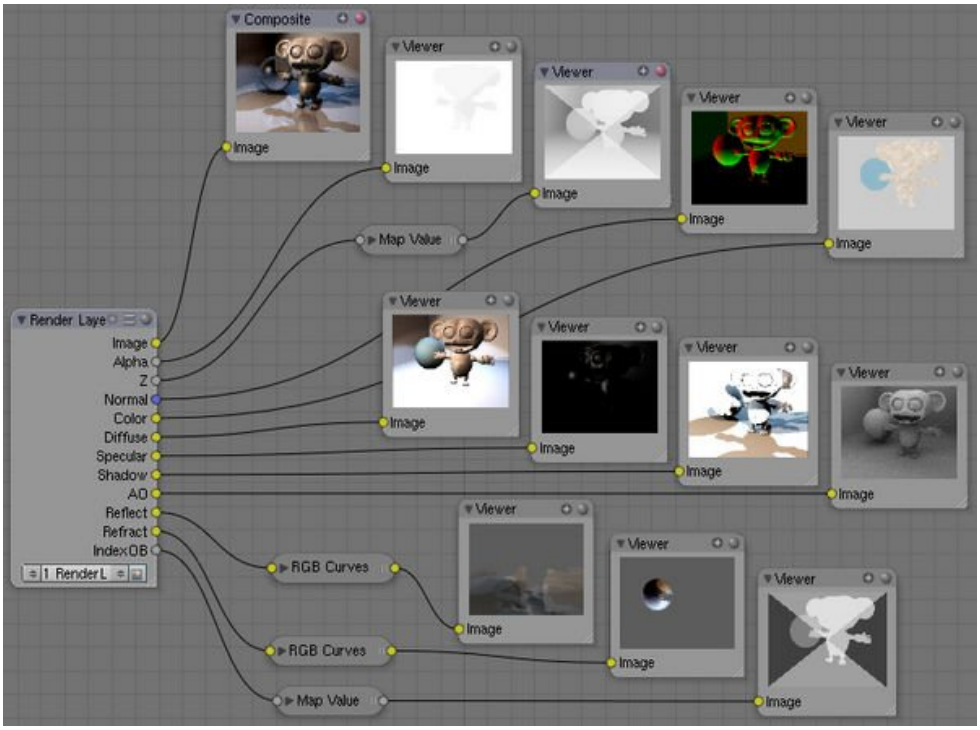
Everything you see in a render must be calculated for the final image. All interactions between objects in your scene, lighting, cameras, background images, world settings, etc. must all be separately calculated in different passes for different reasons, such as calculating shadows or radiosity − in a render, every pixel has been calculated several times to make sure it will show the right colour for the right part of the image. Various things that are calculated in a standard render include:

- *Where are **shadows** cast?* Every light that causes shadows must be 'raytraced', where almost every possible angle from the light is calculated as having a line coming from it, then when that 'line' hits an object, it calculates how much light hit that object, and therefore how bright it should appear.
- *How is **ambient** light in the environment blocked (**occluded**) by objects in the scene?*
- *How is light **reflected** off mirrored surfaces?* Like shadows, lines are calculated, except this time they come from the camera and bounce of mirrored surfaces, so that when these lines hit an object, the engine calculates that this is what the camera should see
- *How is light bent (**refracted**) as is passes through transparent objects?* Does it go straight through? Does it bend? If so, at what depth in the object?
- *What designated **objects** are in the scene, and what is their outline?* Should the object appear blurred, should it appear in sharp focus?
- *How fast is something moving (**velocity**)?* Should it appear blurred with our framerate or is it slow enough to still be focused on properly?
- *How far away from the camera are objects' surfaces (**Z−depth**)?* Can the object's surfaces be seen at all, or are they being blocked by another object's geometry?
- *Does an object have a **normal** vector (bumpmap)?* Do shadows and apparent geometry need to be calculated for any objects?
- *Is there any specularity?* Are objects with textures such as metal shiny at all?

Renderer Rewrite

Starting with Blender 2.42, the render engine was rewritten. See [Unified Renderer](#) if you are using an old version of Blender.

The answer to each of the above questions is an image or map, shown below:

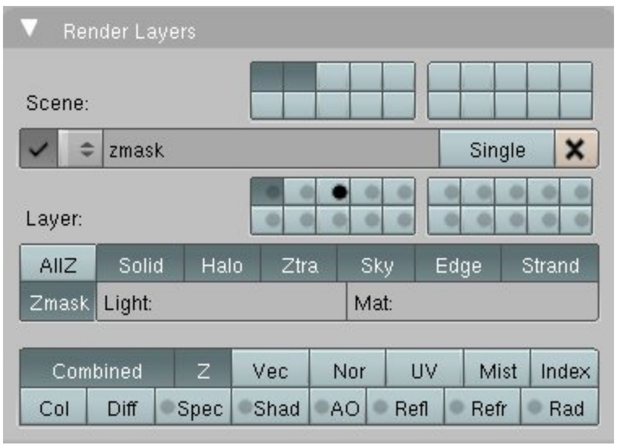


Each RenderPass puts out an image or a map.

RenderPasses that produce Images can be directly viewed in a viewer, or, if they are the only pass that is rendered, saved as the render image. If the pass is enabled, it can be saved in a multilayer OpenEXR format.

If the RenderPass output is not an image but is a map, it needs to be translated into something that we can see. For example, the Z−depth map is an array of values that specifies how far away from the camera each pixel is; values range between +/−3,000,000 blender units or so. The intermediate node you see above, between the RenderLayer output socket and the Viewer node input socket (such as Map Value) does this translation or scaling. You must use that specific kind of translation node to get good results if you intend on operating on that map as an image. You must then, after making any adjustments, run the map back through that node to re−scale it back to original before saving.

Selecting RenderPasses



Each of the above images can be selected for culling as a separate pass. The only reason for culling it out is that you want to adjust it somehow.

Some of them must be enabled and used within your scene (and not just selected in the RenderLayer panel) in order to show anything. For example, if you do not have any lights in your scene, or those lights have been set to not cast shadows, or objects in the limelight do not have materials which have been set to receive shadows, the Shadow pass will be blank; there's simply nothing to show you. If you have not enabled Ambient Occlusion in your world environment settings, the AO pass will be blank, even if you select it here.

To save time and disk space, you have to tell Blender each of the passes to render in the RenderLayers panel:

- **Combined**: This renders everything in the image, even if it's not necessary ("The whole enchilada" so to speak). This is all the options below.
- **Z**: the Z−depth map; how far away each pixel is from the camera. Use for Depth−Of−Field (DOF). The depth map is inverse linear (1/distance) from the camera clip start.
- **Vec**: Vector; how fast things are moving. Use with Vector Blur
- **Nor**: Calculates lighting and apparent geometry for a bumpmap (an image which is used to fake detail on an object) or for changing the apparent direction of light falling on an object.
- **UV**: allows texturing after rendering. See UV node.
- **IndexOb**: masks selected objects. See MaskObj node.
- **Col**: the basic colors adjustment in the image
- **Diff**: Diffuse; main colors before being shaded
- **Spec**: Specularity adjustment
- **Shad**: Shadows cast. Make sure shadows are cast by your lights (positive or negative), and received by materials. To use this pass, mix multiply it with the Diffuse pass.
- **AO**: Ambient Occlusion. Make sure it's turned on in your environment and that RayTracing is enabled.
- **Refl**: Reflection off mirrors and other reflective surfaces (highly waxed white floors, for example). Mix Add this pass to Diffuse to use it.
- **Refr**: Refraction of colors through transparent meshes. Mix Add this pass to the Diffuse pass to use it.
- **Rad**: Radiosity; colors emitted and cast by bright objects spill onto other objects

When you enable a pass, the appropriate socket on the RenderLayers node shows up like magic, and can be used as shown in the example above.

De−Selecting Render Passes

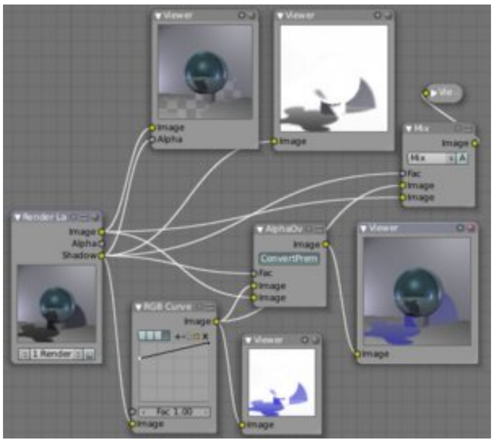
When you Ctrl LMB CTRL−click on a pass, the particular pass will be excluded from the combined pass. Excluded passes are marked by a black dot next to their name.

Using RenderPasses

Quite a bit of information about using some of the passes is discussed elsewhere:

- Image: Since this is the main product, all of Blender uses it.
- Alpha: See the AlphaOver node and all of the Matte nodes
- Z: See the Defocus node
- Vec: See the Vector Blur node
- Normal: See the Normal node

Recoloring Shadows



Let's run the Shadow buffer through a colorization noodle, then recombine it and all your shadows will be artificially colored. Lots of threads in this noodle shown to the right, so let's walk through it. On the left is the RenderLayer input. In the scene, we have a reflective ball on a pedestal standing in front of a backdrop. Everything (except the ball) is gray. We use a standard four light rig: backfill placed high, two sidefills at ground level, and a key light above and to the left of camera. Suzanne, a monkey shaped geometry, is standing in front of the key light, so her shadow is cast into the scene on the floor. The ball casts shadows onto the backdrop and floor. The top two viewers show you the image output using the Shadow as the Alpha channel, and the node next to it just the Shadow channel. Where the Shadow is dark, the image in the left viewer is transparent. We have used the Shadow to cut out parts of the image.

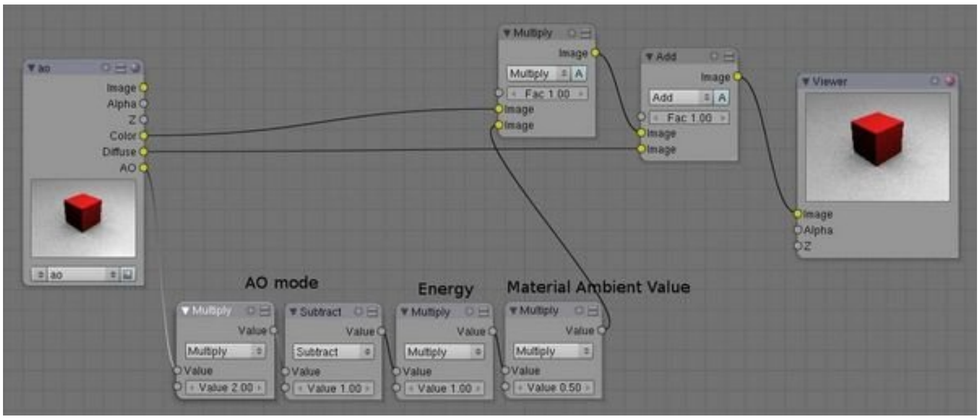
We then take the shadow through an RGB Curve, which is set to magnify just the Blue by 75%; so a gray shadow of (R:40, G:40, B:40) becomes (R:40, G:40, B:40x1.75=70). That blue−tinged shadow is shown in the bottom viewer. Now we have two options: AlphaOver and Mix. For either option:

- Use the Shadow map as a Factor
- Feed the Blue Shadow to the Top Socket
- Feed the core or base image to the Bottom Socket

The resulting image is the same in either case; a blue shadow. Note that Suzanne's reflection is not blue; there's a different RenderPass for that.

You could just as easily swap in another image entirely; for example, the shadow map from another renderlayer. You can even take an image from another project entirely and use that instead (using the Image Input node), to get a different effect (for example, an effect similar to a Star Wars poster, where Anakin Skywalker already casts the shadow of Darth Vader)

Compositing Ambient Occlusion



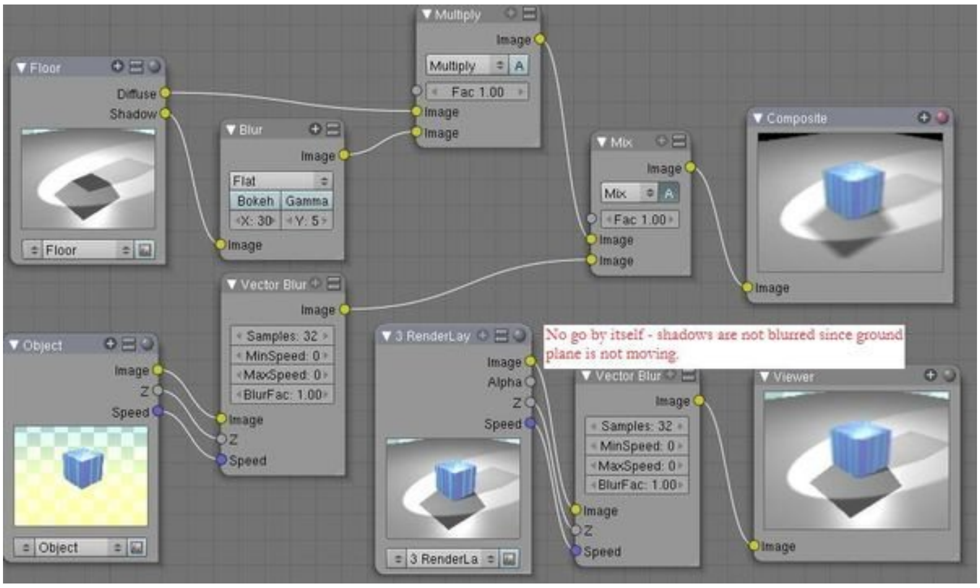
AO is a geometry−based dirt shader, making corners darker. It is separately enabled in the World settings and computed as a separate pass. When enabled, it has one of three Modes (Add, Subtract, Both), and variable Energy level (which changes the intensity of the shading). The third variable is the amount of Ambient light that the material receives. If it does not receive any, then ambient occlusion does not affect it. Based on these variables, Blender computes an AO pass. If you call it out as a separate pass and wish to composite it back into your image, you will need to enable the Color and Diffuse pass as well.

To configure your noodle, consider the example image above.

1. . First, depending on the AO mode do one of the following: If AO mode is Add: directly use the AO pass. If AO mode is Sub: Calculate AO − 1, or if AO mode is Both: Calculate 2*AO − 1
2. . Multiply the output of Step 1 with the AO energy level
3. . Multiply the output of Step 2 with the material's ambience value. If you have materials which receive different ambience light levels (0.5 is the default), one would have to create an ambience map based on Object ID)
4. . Multiply the output of Step 3 with the color pass
5. . Add the output of Step 4 to the diffuse pass

If shadows, colored ambient light, specularity, reflections, and/or refractions are involved they have to be added to the diffuse pass before adding the converted AO pass.

Vector Blurring Shadows



When using Vector Blur, instead of Motion Blur, objects in motion are blurred, but objects at rest (with respect to the camera) are not blurred. The crossover is the shadow of the object in motion. Above, we have a cube in motion across a ground plane. If we just ran the combined pass through Vector Blur, you can see the result in the lower right−hand corner; the box is blurred, but its shadow is sharply in focus, and thus the image does not look realistic.

Therefore, we need to separate out the diffuse and shadow passes from the floor by creating a "Floor" renderlayer. That renderlayer has Diffuse and Shadow passes enabled, and only renders the floor object (layer 2). Another renderlayer ("Cube") renders the Z and Vector passes, and only renders the cube (on layer 1). Using the Blur node, we blur the shadow pass, and then combine the diffuse and blurred shadow by multiplying them together in a Mix Multiply node; we then have a blurred shadow on a crisp ground plane. We can then mix the vector−blurred object to provide a realistic−looking image.

Conclusion

Render Passes can be manipulated from Blender 2.43 to give you almost complete control over your final image. Causing objects to cast shadows that aren't really their shadows, making objects appear out of focus or sharply in focus like a real camera, manipulating colours just for final post−processing or just reconfiguring your render passes to save render time, are all things which you might wish to manipulate the render engine for.

Happy Blending

Previous: Manual/Render Layers	Contents	*Next:* Manual/Render Two−Point

 Perspective (Architectural) Rendering

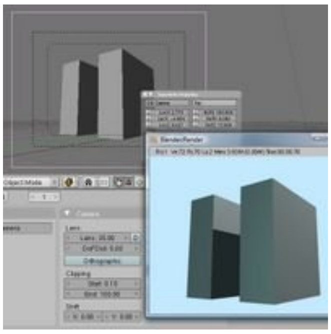
User Manual: Contents | Guidelines | Blender Version **2.43**

Perspective (Architectural) Rendering

When you press F12 and get your render, you see an image as seen through the camera's "perspective". Like how you can *view* your model in 3D View from the top, front, side, or user perspective, you can *render* your object from different perspectives. This perspective takes into account the lens size, type, and offset in giving you that picture. Each perspective uses a different number of vanishing points. If you look at a 3D image of a cube, you will see three kinds of edges: vertical, horizontal, and depth. If all of the vertical edges are exactly parallel, there is no vanishing point for them. If however, they are not parallel, if you extended them by continuing them with a ruler, they would at some point intersect. That point is called the vanishing point.

For special purposes, different kinds of render cameras can be set up to give you different perspectives. For reasons discussed below, you may wish to limit the number of vanishing points, especially for architectural purposes. Architects and drafting people are responsible for rendering the object or building with true dimensions and true relative proportions. If you look at that example render, the building looks all sorts of distorted, like it had been made of mud and was collapsing. If you told a builder to build that, you would end up with a building that actually had leaning walls and rooms that were narrower at the top. Way back in the old Greek days, when they started building tall columns, they built them thicker at the top than at the bottom, so that when viewed looking up, the two sides would look straight up and down. Then they even started narrowing the columns at the top to give the illusion that the building was taller and would look higher. During the Renaissance, the concept of using vanishing points in art evolved. Blender offers a few tricks of its own to let you do the same.

Three Point Rendering

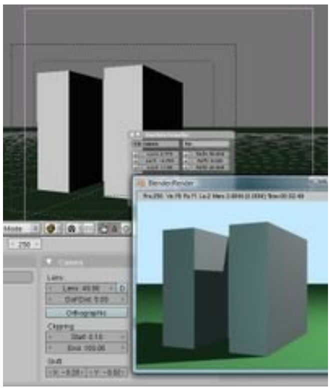


Normal Three Point Render

When looking at or rendering a picture of a high building from ground level off to one side, and aiming up, using the normal 35mm camera, you get 3 point perspective. If you laid a ruler along the vertical lines, you would see that they converge to a point above the building. The horizontal lines are converging off to one side (the left in this example), and depth (receding) lines are converging to a different third point (somewhere off to the lower right in this example). Hence the name 3−point rendering − there are three vanishing points.

This is reality, and there is nothing wrong with that. When you next step outside and look at a tall building, this is what you actually see. However, your mind knows that the building is square, and can adjust your perception of the building so that you are not scared that the building is going to fall over.

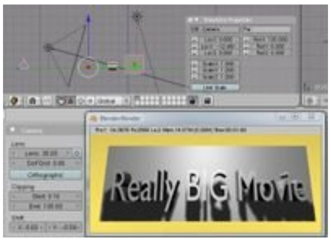
Two Point Rendering



Two Point Vertical Render

Normal architectural rendering is called two point rendering; when vertical lines are parallel, and horizontals, if followed out to the side, converge on one point, and receding or depth lines converge to a second point. Architects often like this Two−Point rendering, so that the sides of their buildings are completely vertical and don't appear to be falling inward. This is also quite nice for compositions and schematics, given that the lines of the paper you print on and the screen you view with are also straight.

Previously to get a 2−point perspective, you had to aim the camera level to the horizon, however this resulted in the top half of the building being cut off and the horizon being in the exact middle, which looks very boring. Architectural photographers use 'shift lenses' to solve this problem. Shift lenses shift the image to another place on the film.



Two Point Horizontal Render

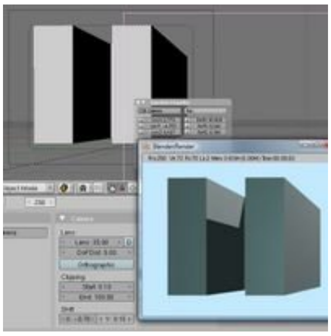
This technique works well for high buildings as well as for normal sized objects. Most of the time, the two vanishing points are horizontal and depth lines, with the vertical lines parallel. However, some titles are done with the horizontal lines parallel, and the vertical and depth lines having the vanishing point. This dramatizes and exaggerates the massiveness and height of the title. To get this effect, position the camera at ground level, centered, angle the camera upward, and shift the render passpartout down. In the example, the camera is rotated 30 degrees upward, at ground level with the title. A bright key light with a short falloff provides dramatic lighting that is bright in the middle and falls off toward the sides, further enhancing the depth.

To achieve 2−point rendering:

- Use a short wide angle lens camera, say with a Lens Size of 10 mm placed close to the building, or a long lens farther away from the building. These differences affect the depth of the building render, with longer lenses making the building appear thinner and less dramatic or distorted. The example uses a 40mm lens.

- Position the camera off to one side of the object, vertically halfway up the building to minimize distortion of the vertical building edges. You may alter this vertical (Z value) position to be slightly higher than ground level or higher than the top (if you want to see the top of the object or building). To show the front bottom corner of the building jutting out, raise up the camera.
- Angle the camera to be looking away from the building and directly level at the horizon − not pointed up or down (note the 20 degree Z angle in the example). This should make the vertical lines parallel. The more the camera looks at the object, the closer the vanishing point for the horizontal lines, and perceived depth will increase as that vanishing point gets closer as well.
- You may have to angle the camera slightly down (just 1 degree or so) so that vertical lines appear vertically up and down, both near and far. If the lines are curved, use a longer lens. With your 3D View set to Camera view, use the passpartout or pixels on your monitor to determine vertical.
- Move the camera toward/away from the object until it appears near a corner of the render and is the right size.
- Adjust the Shift: X and Y settings until your object is positioned properly.

One Point Rendering



One Point Render

One point rendering is where vertical and horizonal lines are parallel, and depth lines converge at one point. Architects really like these renders, since the front−facing faces are true and square, and the building recedes off into the distance so that it looks like it has some depth. If the camera is placed at ground level, even with the bottom of the building, it really looks dramatic but orderly in a weird sort of way. Title graphics are sometimes rendered this way.

To get 1−point (1pt) renders,

- To get more dramatic depth lines, use a short wide angle lens camera, say with a Lens Size of 10 mm, very close to the building. For a more normal appearance, stick with the 35mm lens.
- Position the camera off to one side of the object, slightly higher than the top (if you want to see the top of the object) or at ground level (the example image has the camera almost at ground level). If you position the camera *below* ground level, the bottom depth lines and horizontal lines will merge up (become congruent) for a *very dramatic* effect.
- Angle the camera looking straight back, perpendicular to the true face. Vertical lines should be parallel. Rotate the camera on the Z axis *slightly* toward the object until the horizontal edges are also parallel. Technically, you are correcting for parallax (just a casual line to drop on your girlfriend to impress her). The example has the camera rotated 0.5 degrees toward the object.
- Move the camera toward/away from the object until it appears at the proper size relative to your passpartout.
- Adjust the Shift: Y settings until the bottom of the passpartout (or title line if you want to show some approach ground in front of the building) is even with the bottom of the building. Adjust the X setting

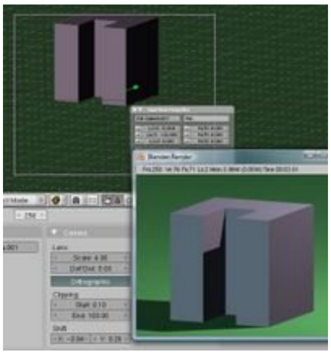
until the building is centered (or slightly offset from center for artistic appeal, or to show the parking lot next to it) as shown.

In the example screenshot, the Lens is 35, X is negative and Y is positive. The camera is off to the right of the object, even with the bottom of the building. If X & Y were zero, the building would have appeared off camera, in the upper left−hand corner of the passpartout.

Parallel Horizontal Edges:

You can use the lines of the passpartout as a guide in rotating the camera to determine when the horizontal edges are parallel.

Zero Point (Orthographic) Rendering



Orthographic Render

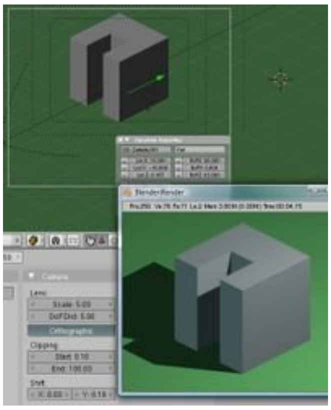
Zero point rendering is where vertical, horizontal AND depth lines are all parallel, and is commonly rendered at 45 degree, 30 degree, or 60 degree angles. With all of those sets of edges parallel to each other within that set, there are no vanishing points. The example shows that same building rendered at 45 degrees from all angles. Note that the vertical lines are parallel to each other, the horizontals, and the depth lines are parallel to each other. From this, it is very easy to see that the left top edge of the building is the same length as the right top edge, and that the building is as deep as it is wide and high; if you measured the edges with a ruler, they would all be the same. Orthographic rendering gives a true mathematical render of the shape of the object. An Orthographic perspective is what you see in the User View of a 3D window (if View−>Orthographic is turned on).

To get an Orthographic render:

- Enable Orthographic in the Camera panel. This makes at least one face to be true to the camera.
- Point the camera at the object
- Position the camera or alter the Scale so the object is the desired size

With Orthographic cameras though, Lens size is irrelevant, since light rays do not converge to the camera from a field of view. They come in parallel, and so you can only Scale the camera size to take in more or less of that huge plane. Note that Shift X & Y are zero, and that the camera is positioned perfectly off at a 45 degree angle to the object/building, and is rotated exactly 45 degrees to face the building. Thus, the near edge is aligned with the back edge (since the object is square). Orthographic renders are usually made at 30, 45, or 60 degree angles to the object. Specific measurements are left to reader using triangle math.

Isometric Rendering



Isographic Render

While we are at it, we might as well cover Isometric rendering, which is a very specific type of orthographic render very often used in drafting and third−person computer games. In Isometric renders, you want your depth lines and your horizontal lines to be at 30 degrees off horizontal, and your vertical lines to be, well, vertical. Some complicated vector calculus in the wikipedia gives us a convenient shortcut. To get Isometric Renders:

- Make your camera Orthographic
- Add a "Track To" constraint (Object F7 context, Constraints panel) to the camera for it to Track To the object (type the name in the Target OB: field), using To: −Z and Up Y.
- Position your camera so that it is 45 degrees in the XY plane from your object, and raised at a 30 degree angle. If your object is at XYZ (0,0,0), then your camera should be at (10, −10, 10), or for a view from the left side, (−10, −10, 10)
- Adjust the Scale of the camera (Editing F9 context, Camera panel) so that the object fits within the passpartout
- Adjust the Shift: Y value so that the object is centered in the render.

Previous: Manual/Render Passes	Contents	*Next:* Manual/Rendering Your Work For Video

User Manual: Contents | Guidelines | Blender Version **2.43**

Preparing your work for video

Once you have mastered the trick of animation you will surely start to produce wonderful animations, encoded with your favourite codecs, and possibly you'll share them on the Internet with the rest of the community.

Sooner or later you will be struck by the desire of building an animation for Television, or maybe burning your own DVDs. To spare you some disappointment, here are some tips specifically targeted at Video preparation. The first and principal one is to remember the double dashed white lines in the camera view! If you render for PC then the whole rendered image, which lies within the *outer* dashed rectangle will be shown. For Television, some lines and some part of the lines will be lost due to the mechanics of the electron beam scanning in your TV's cathode ray tube. You are guaranteed that what is within the *inner* dashed rectangle in camera view will be visible on the screen. Everything within the two rectangles may or may not be visible, depending on the given TV set that your audience watches the video on.

The rendering size is strictly dictated by the TV standard. Blender has three pre−set settings for your convenience:

- PAL 720x576 pixels at 54:51 aspect ratio.
- NTSC 720x480 pixels at 10:11 aspect ratio.
- PAL 16:9 720x576 at 64:45 aspect ratio, for 16:9 widescreen TV renderings.
- HD 1920 x 1080 pixels at 1:1 aspect, that operates in a downscaled mode of 720 horizontal scan lines interlaced.

Aspect Ratio

TV screens do *not* have the square pixels which Computer monitors have; their pixels are somewhat rectangular, so it is necessary to generate *pre−distorted* images which will look bad on a computer but which will display nicely on a TV set.
If you render your animation at 1600x1200 resolution, and then burn a DVD, your image will not be clearer or crisper on the TV; in fact the DVD burning sofware will have had to downsize your images to fit the resolutions shown above, and you will have wasted about 4x disk space and render time.

Colour Saturation

Most video tapes and video signals are not based on the RGB model but on the YCrCb model: more precisely, the YUV in Europe (PAL), and the YIQ in the USA (NTSC), this latter being quite similar to the former. Hence some knowledge of this is necessary too.

The YCrCb model sends information as 'Luminance', or intensity (Y) and two 'Crominance' signals, red and blue (Cr and Cb). Actually a Black and White TV set shows only luminance, while colour TV sets reconstruct colour from Crominances (and from luminance). The contruction of the YCrCb values from the RGB ones takes two steps (the constants *in italics* depend on the system: PAL or NTSC):

- First, the Gamma correction (*g* varies: 2.2 for NTSC, 2.8 for PAL):
    - $R' = R^{1/g}$
    - $G' = G^{1/g}$
    - $B' = B^{1/g}$
- Then, the conversion itself:

- ♦ Y = 0.299R' + 0.587G' + 0.114B'
- ♦ Cr = a_1(R' − Y) + b_1(B' − Y)
- ♦ Cb = a_2(R' − Y) + b_2(B' − Y)

Whereas a standard 24 bit RGB picture has 8 bits for each channel, to keep bandwidth down, and considering that the human eye is more sensitive to luminance than to chrominance, the luminance signal is sent with more bits than the two chrominance signals. This bit−expansion results in a smaller dynamic of colours, in Video, than that which you are used to on Monitors. You hence have to keep in mind not all colours can be correctly displayed. Rule of thumb is to keep the colours as 'greyish' or 'unsaturated' as possible, this can be roughly converted in keeping the dynamics of your colours within 80% of one another. In other words, the difference between the highest RGB value and the lowest RGB value should not exceed 0.8 ([0−1] range) or 200 ([0−255] range). This is not strict, something more than 0.8 is acceptable, but an RGB display with a color contrast that ranges from 0.0 to 1.0 will appear to be very ugly (over−saturated) on video, while appearing bright and dynamic on a computer monitor.

Rendering to fields

Mode: All Modes

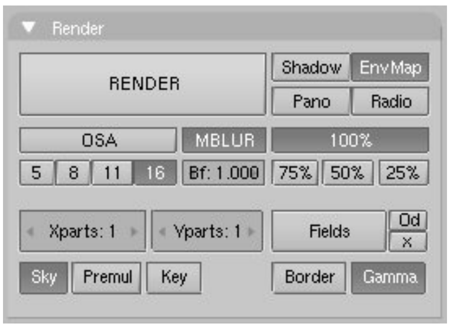
Panel: Render Context Render

Hotkey: F10

Description

The TV standards prescribe that there should be 25 frames per second (PAL) or 30 frames per second (NTSC). Since the phosphorous of the screen does not maintain luminosity for very long, this could produce a noticeable flickering. To minimize this TVs do not represent frames as a Computer does ('progressive' mode), but rather represents half−frames, or *fields* at a double refresh rate, hence 50 half frames per second on PAL and 60 half frames per second on NTSC. This was originally bound to the frequency of power lines in Europe (50Hz) and the US (60Hz). In particular fields are "interlaced" in the sense that one field presents all the even lines of the complete frame and the subsequent field the odd ones. Since there is a non−negligible time difference between each field (1/50 or 1/60 of a second) merely rendering a frame the usual way and splitting it into two half frames does not work. A noticeable jitter of the edges of moving objects would be present.

Options

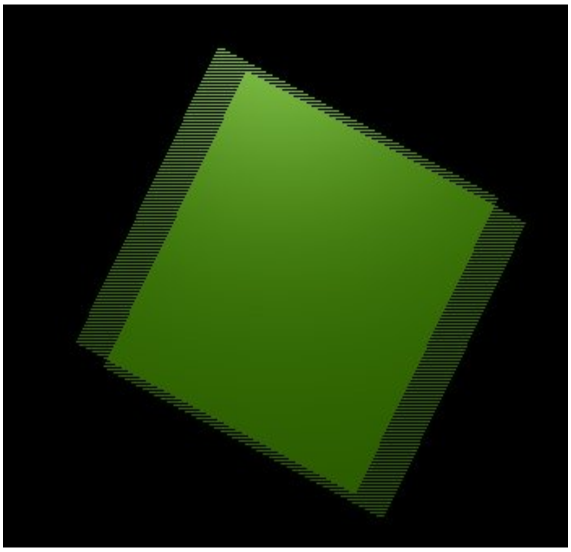


Field Rendering setup.

Fields

Enable field rendering. When the

Fields button in the Render Panel is pressed (*Field Rendering setup.*), Blender prepares each frame in two passes. On the first it renders only the even lines, then it *advances in time by half a time step* and renders all the odd lines.



Field Rendering result.

This produces odd results on a PC screen (*Field Rendering result.*) but will show correctly on a TV set.

Odd
: Forces the rendering of Odd fields first

X
: Disables the half−frame time step between fields (x).

Setting up the correct field order

Blender's default setting is to produce Even fields *before* Odd fields, this complies with European PAL standards. Odd fields are scanned first on NTSC. Of course, if you make the wrong selection things are even worse than if no Field rendering at all was used.

Fields and Composite Nodes

Nodes are currently not field−aware. This is partly due to the fact that in fields, too much information is missing to do good neighborhood operations (blur, vector blur etc.). The solution is to render your animation at double frame rate without fields and do the interlacing of the footage afterwards.

Previous: Manual/Render Two−Point	Contents	*Next:* Manual/YafRay

User Manual: Contents | Guidelines | Blender Version **2.45**

Blender produces all the information needed to render a scene. While it has its own internal rendering engine, you can export or link to external renderers for image computaiton. Some external renderers include:

- [Yafray]
- [Yaf(a)ray]
- [Indigo]
- [Kerkythea]
- [PovRay]
- [MentalRay]
- [Sunflow]
- [Luxrenderer]

Of these, Blender is tightly integrated with Yafray, and an overview and quick start guide is provided here. For more detailed information, consult the *[Yafray web site].

Rendering with Yafray

Mode: Any Mode

Panel: Render Context Render / Render Context Yafray

Hotkey: F10

Description

Yafray, as the lengthened version of its name (Yet Another Free RAYtracer) suggests, is a free, XML speaking, cross platform raytracer developed by the Yafray team.. It works with many 3D modelling applications (with Wings and Aztec serving as examples), but the focus on this document shall fall upon it's use with Blender.

Yafray is currently available (under the LGPL license) for Windows, Linux (via source code compilation, or .deb or .rpm installation), Mac OSX, and Mac Intel; and installation packages, as well as Yafray's source code, can be downloaded here.

Options

All post−2.34 Blender releases have the option to call Yafray in the place of Blender's internal renderer, assuming it's installed. This can be done by switching to the "Render buttons" panel (F10) and selecting Yafray as the rendering engine.

Render Pipeline

When Yafray is used, it is inserted into the pipeline before any compositor or sequencer actions, because it is the renderer, and the compositor and sequencer work on rendered images. The image data given to Yafray is the scene objects, materials, lights, etc. Yafray does not know nor care about render layers and cannot feed Blender's node compositor or sequencer effects, since it takes a completely different approach and cannot produce the different render layers that the Blender internal renderer can. Yafray render frames based on Blender scene data.

To use Yafray with Blender's compositor, render the image using Yafray, and then use the image input node to get that image into the compositor where it can be post−pro. You can then feed that to Sequencer via the Scene strip and Do Composite enabled. To feed the Sequencer directly from Yafray's output, use the Image strip after Yafray has completed the render.

Two other panels should appear once Yafray's selected from this menu, which serve to supply a number of Yafray's options to you (other options are available exclusively as XML code).

Enabling Yafray

XML
: This button, if pressed, will export your scene to a .xml file in your system's 'tmp' directory before Yafray renders it. Useful if you wish to make modifications to the XML file, or render the scene from a command line interface. Should you wish, however, to view Yafray's progress during a render in Blender's render window, it's better to unclick this button.

AutoAA
: This options allows you to toggle between manual and automatic control of anti−aliasing options in the scene. Anti−aliasing is similar to Blender internals OSA, which in effect dictates the accuracy of the edges in the render.

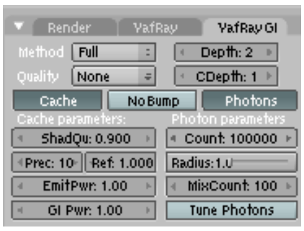
  In cases where you may need to manually control the anti−aliasing options (which becomes necessary if you wish to make use of Yafray's depth−of−field option), it's useful to remember that increasing the amount of samples per pass will increase the accuracy of the edges in the final render; decreasing the amount of samples per pass will, as you'd expect, decrease the accuracy, causing edges in the scene to seem rough, and jagged.

Proc.
: This option allows you to select the number of processors Yafray's allowed to make use of. For those of us who aren't lucky enough to have multiple processors, it's best to leave this option as it's default.

Gam.
: This option allows for manual correction of gamma values in the scene. The default (1) turns this option off.

Exp.
: This option allows for manual adjustment of exposure levels in the scene. A more indepth explanation of this option will come later.

Global Illumination



Global illumination settings

The next tab along, titled "Yafray GI", provides a selection of methods with which Yafray's able to light a scene. Methods available are:

Full
: This method works well for most scenes, most notably indoor scenes, where the use of photons becomes appropriate.

Skydome
: This method is more suited to outdoor scenes.

Cache
: Clicking the **cache** button speeds up rendering by allowing Yafray to be more selective it's distribution of samples. When this button's depressed, Yafray renders a pre−pass to determine the most suitable allocation of samples, before rendering the image itself, increasing the efficiency of the render.

The cache button then reveals three more options.

ShadQu
: This option allows for greater control over the quality of shadows. By increasing this option from its default (0.900), you also increase the number of samples taken in shadowed areas, which in turn not only increases the quality of shadows in the scene, but also increases render times.

Prec
: This option sets the maximum number of pixels per−square without samples. By decreasing this option from its default (10), you increase the number of samples taken in the scene. Decreasing this option also increases render times.

Ref
: This option allows the user to specify the threshold to refine shadows. By decreasing this option from its default (1.000), you invite Yafray to increase the number of passes taken to distribute samples in shadowed areas, thereby increasing the quality of the shadows in the scene, and increasing render times.

Examples

Starting with the default Blender set up, enable Yafray in the Rendering Buttons panel (F10), deselect the XML option in the "yafray" tab, and select the "full" method from the "yafray GI" tab, and set the quality to "low". Then click "Render" (F12).

Console output

Provided the evironment allows it, Yafray should output information to the console window (in Windows, Blender opens along side a console window by default. In GNU/Linux, however, to view the console output, you'll need to start Blender from the console − Usually by typing "blender" into a terminal emulator window).

If you switch to the console after the render's completed, you should (provided the "cache" option's enabled) notice something similar to this:

Console output

Launching 1 threads

Fake pass: [#############]
534 samples taken

Render pass: [#############]

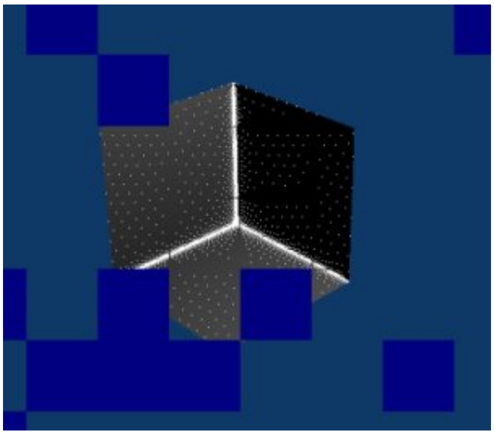
render finished

Output description

The render's split up into two seperate passes. The first, "fake" pass is made as a direct result of the "cache" option being enabled, and it's purpose is to determine the best distribution of samples in the scene (without the cache option enabled, the samples are distributed evenly among the scene). The number of samples is then output onto the next line.

The next pass is the "real" render pass, where Yafray renders the image based on the sample map created in the previous pass.

Render window output



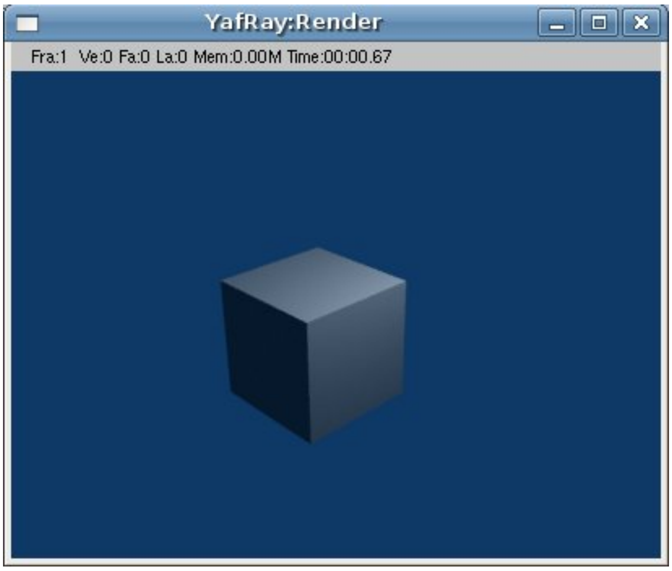
Greater samples in shadowed areas

Now we'll look at the Yafray's output to the render window, during the render.

Provided the XML option is turned off, Yafray will continually update it's visual output to the render window − Much like Blender does. The image to the right was captured during the "fake" pass stage of the render, and the white dots represent the allocation of samples in the scene. Notice how the samples are only placed on areas of the scene that are directly affected by light, meaning that, in the demonstration image, only the parts of the scene with a surface are considered.

This also means that in shadowed areas of the scene, the number of samples is greater.

You can notice that the density of white dots which, as I pointed out earlier, represent the number of samples per pixel in that area of the image, is greater in areas that are likely to be shadowed (in this case, I deleted the vertex of the cube closest to the camera, revealing inside edges, which aren't as exposed to the light).



Basic Yafray render

The rendered image

You'll notice how the cube, despite Blender's default grey material being applied, has been coloured blue. This is because the Full method is affected by the "world" colour of the scene, which, again as Blender's default, is blue. To change this, switch to the "shading" panel (F5), and select the little world icon. To have materials show properly, set the world shader to white.

Selecting the world shader

Notes

Amount of Light

YafRay deals with light completely differently than the Blender Internal Renderer, and apparently light intensity needs to be pumped by large amounts for YafRay. the images reflect a Blender Internal, a Yafray render without Global Illumination (GI), and one with Full GI. As you can see, results vary widely based on the illumination method chosen.

A solution is to use very large Area lamps (Square, 100 Size but Samples at only 4, Energy 10) for softer shadows, in combination with a Sun lamp at much lower Energy value (less than 1.0) if you want a distinct shadow edge. Sun lamps seem to provide much greater intensity than Area lamps in YafRay but the shadow edges are quite harsh.

Try using the Skydome setting for the YafRay GI because with Full GI you may get weird blotchy artifacts that no one seems to know how to remedy, but may be related to the scale of my Blender scene, which is 1BU = 1cm, with a figure built to life−size. You'll be doing something like this as well if you build a scale model to match camera perspectives.

Blender World parameters may include a small AO setting which YafRay does seem to take into account, so you might try adding some in your scene. Also be aware that the World Sky colors (Ho & Ze) are treated as a "hemi" light source, and will color your scene accordingly when using Skydome −− play with these RGB

values to perhaps boost the overall lighting intensity by "filling in" with GI. In the pics below, the World lighting settings were doubled for the render on the right.

Everything seems to need to be boosted for YafRay −− some Materials look very dull unless you "double−up" some of the components (such as by using an image texture twice with "Add"), and the RGB & Shader tab settings are very different from what you would use with the Internal renderer.

You can also adjust the EmitPwr and Exp settings in the YafRay renderer tabs to compensate for the lighting differences. It gets to be quite a juggling act. The plus side is that you are able to get lighting of a much richer character for a scene, so it can be worth the trouble.

SkyDome

Various coloring effects based on World settings

Using the Blender Internal (BI) renderer, the only way to get the world Horizon, Zenith, or Textured color to affect the material color is to use Ambient Occlusion set to Sky Color or Sky Texture; otherwise (without AO) it only affects the color of the background. The only variable to directly affect the final object coloration in Blender Internal is the color of Ambient light, and then each material can receive a specified amount of that ambient light (by default 50%). The color of the ambient light in BI cannot be varied over the height of the image and is applied uniformly to the subject. Ambient Occlusion, based on the settings, affects the color of the model based on its geometry.

In Yafray, however, a key difference is that the color of all of these matter, as shown in the example. The example has the same material (the skin and hair) rendered using different **Horizon and Zenith** colors. Each of these, in effect, change the ambient light cast onto the subject. If the Zenith was darker, as is usually the reality, the tops of the model would be darker than the the lower portions. Using the color of the sky and horizon to affect the lighting of subjects lends a much more realistic blending of a subject into the environment, leading to more photorealistic results.

To achieve the same effect in Blender, you can use Ambient Occlusion, or light your subject with Hemisphere lamps which are the same color as your sky zenith and horizon.

Previous: Manual/Rendering Your Work For Video	Contents	*Next:* Manual/Rendering_From_Command_Line

User Manual: Contents | Guidelines | Blender Version **2.43**

Introduction

In some situations we want to increase the render speed, access blender remotely to render something or build scripts that use blender command line.

One advantage of using command line is that we don't need the X server (in case of Linux) and as a consequence we can render remotely by SSH or telnet.

Note! Arguments are executed in the order they are given!

```
blender -b file.blend -a -x 1 -o //render
```

...Wont work, since the output and extension is set after blender is told to render.

Always position **−f** or **−a** as the last arguments.

Description

Usage: blender [−b <dir><file> *[−o <dir><file>*][−F <format>][−x *[0|1]*][−t <threads>][−S <name>][−f <frame>][−s <frame> −e <frame> −a]] [[−P <scriptname> [−− <parameter>]]

Render options:

```
 -b <dir><file> Render <file> that is inside the <dir> without load the UI
   -P <filespec>Run the specified Python script (filename or Blender Text)
   -S <name>     Set scene <name>

   -f <frame>    Set frame <frame> to render and save it (Don't use together with -a)
   -j <number>   Render every x frames (jump frames by this number)

   [-s <frame>] [-e <frame>] -a
                 Set start frame (-s), end frame (-e) or both.
                 The order is important and it's possible to use only the -s or the -e
                 alone.

   -o <dir><file>
                 Set the render path and file name.
                 Use // as <dir> to use the path render relative to the blend file.
                 Use # in the filename to be replaced with the frame number
                   eg: blender -b foobar.blend -o //render_# -F PNG -x 1

   -F <format>   Set the render format, Valid options are..
                 TGA IRIS HAMX FTYPE JPEG MOVIE IRIZ RAWTGA
                 AVIRAW AVIJPEG PNG BMP FRAMESERVER
      (formats that can be compiled into blender, not available on all systems)
       HDR TIFF EXR MPEG AVICODEC QUICKTIME CINEON DPX

   -x [0|1]      Set option to add the file extension to the end of the file, 0 means
                 no and 1 means yes.

   -t <threads> Use amount of <threads> for rendering
```

Animation options: (Used when pressing the play button, blender behaves like a movie player)

```
-a <file(s)>  Playback <file(s)> (Don't use together with -b!)
  -p <sx><sy> Open with lower left corner at <sx>,<sy> (Doesn't work on win)
  -m          Read from disk (Don't buffer)
```

Window options:

```
-w             Force opening with borders (default)
-W             Force opening without borders (Linux/Unix Only)
-p <sx> <sy> <w> <h>  Open with lower left corner at <sx>, <sy>
                      and width and height <w>, <h>
```

Game Engine specific options:

```
-g fixedtime          Run on 50 hertz without dropping frames
-g vertexarrays       Use Vertex Arrays for rendering (usually faster)
-g noaudio            No audio in Game Engine
-g nomipmap           No Texture Mipmapping
-g linearmipmap       Linear Texture Mipmapping instead of Nearest (default)
```

Misc options:

```
-d             Turn debugging on
-noaudio       Disable audio on systems that support audio
-h             Print this help text
-y             Disable script links, use -Y to find out why its -y
-P <filename> Run the given Python script (filename or Blender Text)
-R             Register .blend extension
-v             Print Blender version and exit
```

Examples

Render a picture

```
# blender -b file.blend -o //file -F JPEG -x 1 -f 1
```

- **−b file.blend** : File .blend to render
- **−o //file** : Directory + Target image file
- **−F JPEG** : JPEG image format
- **−x 1** : Adds an extension .jpg to the file name
- **−f 1** : Render frame 1

Render a movie

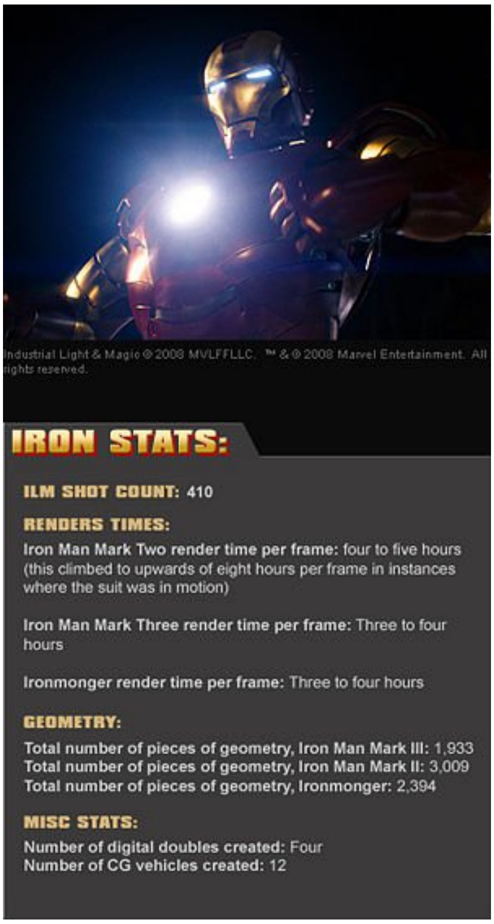
```
# blender -b file.blend -x 1 -o //file -F MOVIE -s 003 -e 005 -a
```

- **−b file.blend** : File .blend to render
- **−x** : Adds an extension .avi to the movie
- **−o //images/file** : Directory + Target image file
- **−F MOVIE** : This saves a .AVI movie with low compression
- **−s 003 −e 005 −a** : Set start frame to 003 and end frame to 005. ***Important:*** *You can use −s or −e, but if they're not in order, they'll not work!*

Previous: Manual/YafRay | Contents | *Next:* Manual/Compositing

User Manual: Contents | Guidelines | Blender Version **2.46**

Rendering Big Hairy Scenes or Speeding Up Rendering



"A watched pot never boils" is the old saying, but you may wonder why your render takes so long to create, or worse, crashes mid−way through! Well, there is lots going on and lots you can do to speed up rendering or enable a complicated render to complete. Also, it is possible to render a very complicated scene on a mediocre PC by being "render−smart". Here's a "top ten" list of things to do or not do in order to speed up rendering or even avoid crashes on scene render. Some options may decrease the quality of your render, but for draft renders you may not care.

If you get the message "Malloc returns nil", in plain english that means the memory allocator tried to get more physical memory for Blender but came back empty−handed. This means that you do not have enough memory available to render the scene, and Blender cannot continue. You will need to do one or more of the following tasks on this page in order to render.

Hardware Improvements

1. Get more RAM up to your PC's (motherboard and operating system) limit. Presently, Blender can use up to 8GG (giga−giga) of physical memory (64−bit address space), but most PC's can only handle 2G of RAM.
2. Upgrade your CPU to a multi−core/multiprocessor
3. Upgrade your OpenGL video drivers
4. Get a faster memory bus
5. Get faster memory, up to yor PC's motherboard limit. 667MHz memory is 30% slower than 800MHz.

6. Use or set up a render farm using all available PC's in your house, or use a render farm such as BURP.

Operating System Configuration

1. Increase the processing priority through your OS
2. Increase your swap file space used by the OS for memory swapping. Also called virtual memory pagefile size, up to the size of your physical memory.
3. Upgrade to a 64−bit operating system
4. Exit or stop any background processes, like virus scanners, BOINC, Real, even "inactive" ones like Quicktime, as they can randomly start up to look for updates. They also take up RAM.
5. Disable network connections to stop random pinging traffic and refresh traffic
6. Stop listening to internet radio and close web browsers, especially any multi−media (audio/video/game) sites.
7. Close down all other running applications, like Word
8. Exit all TSRs (those icons on your system tray) and widgets, any background processes, and virus scanners

Blender Choices

1. Increase the MEM Cache Limit in the User Preferences System & OpenGL tab
2. Upgrade to an optimized Blender build, especially if you have a modern chip that supports SSE2 − render times are **30% faster** using an optimized build.
3. Switch to an Orthographic camera, and render your own "parts" of the scene as separate images, and then paste those parts together in GIMP. An old trick in making your own panoramic with a real camera is to take three or so pictures of a very wide (beach sunset) scene, where you take one picture, rotate to the right, snap another, then another, and when you get the pictures developed, you overlap them to make a very wide landscape image. Do the same in Blender; render out one shot to a file, then move the camera to look at a different area of the scene, and render that shot. Each shot will be of a smaller area and thus take in fewer polygons/faces. Be sure that when you position your camera that you snap overlapping shots, so that you can then match them up. If you don't want to use GIMP, you can use compositing nodes and the Translate node to match them up in Blender.
4. Minimize the render window (and Blender if rendering to an internal window). ATI users report dramatic speedup on a per frame basis, which adds up over frame range.
5. Use the Big Render script to render sub−sections of the overall image, and then paste them together.

Scene and Specific Objects

1. Remove lamps, or move them to un−rendered layers, or tie them to layers
2. Turn off some lamp's shadows, using only one or two main sun lamps to cast shadows. A few "shadows only" lights will render faster than every light having shadows on.
3. Use Buffer Shadows rather than Raytraced Shadows
4. Bake your shadows using Render Baking Full Render bake on surfaces that do not move. Using that texture for that mesh, then disable shadows for that material.
5. Simplify meshes (remove polygons). The more vertices you have in camera, the more time it takes to render.
6. Remove Doubles, or use the Decimator mesh edit feature.
7. Remove Subsurf and Multires modifiers
8. Delete backsides of meshes (removing unseen vertices)
9. Render just a few objects at a time; in the beginning of your project, render the background objects and set that will not change and will always be in the background.

10. Put the buildings on another layer, and through renderlayers, don't render them. Then composite them back in later.
11. Make the camera static so that you can better accomplish the above two ideas.
12. Avoid use of Area lights
13. Make materials Shadeless
14. Render Bake AO and textures, and then make those materials Shadeless
15. Decrease the Clip distance for spot lights
16. Decrease the Clip distance for the camera
17. Turn off world AO
18. Turn off Material SSS
19. Use smaller image textures. A 256x256 image takes only 1% of the memory that a 2k image does, often with no loss of quality in the ulitmate render.
20. Reduce Subsurf. Each level quadruples (4x) the number of faces from the previous level.
21. Reduce Multires
22. Make a matte render of background objects, like buildings, and put the image of them on a billboard in the scene instead of the object themselves. This will reduce vertex/face count

Render Settings

- Output Panel
    1. Disable Edge rendering.
    2. Save Buffers
    3. Render to an Image Editor window, not a popup Render Window.
    4. Use multiple Threads on a multi−core CPU (with multiple Parts).
- Render Layers Panel
    1. Render only the Layers of interest.
    2. Render with all lights set to one simple spot (enter its name in the Light: field)
    3. Render with one material override (enter its name in the Mat: field)
    4. Disable unnecessary Render Passes, such as Z, or only render the pass of interest, such as Diffuse.
- Render Panel
    1. Turn off Shadows
    2. Turn off Environment Mapping
    3. Turn off Panoramic Rendering
    4. Turn off Raytracing
    5. Turn off Radiosity
    6. Turn off SSS Subsurface Scattering
    7. Turn off or lower oversampling/aliasing OSA
    8. Turn off or lower Motion Blur
    9. Render in Parts. This will also allow you to render HUGE images on a weak PC.
    10. Render at a percentage size of your final resolution (like 25%)
    11. Turn off Fields rendering.
    12. Use Border rendering to render a subset of the full image.
- Anim Panel
    1. Decrease the frame count of the animation (and use a lower framerate for the same duration of animation). For example, render 30 frames at 10 frames per second for a 3−second animation, instead of 75 frames at 25 frames per second.
- Bake Panel
    1. Bake Full Render − create a [UV Texture that colors the objects] based on materials, and then use that UV Texture shadeless instead of the material − 7x faster
    2. Bake Ambient Occlusion only
    3. Bake textures for objects

◊ Baking Normals or Displacement does not speed up render time, and are used for other things.

- Format Panel
    1. Render at a lower resolution. Smaller pictures take less time to render.
    2. Choose a faster CODEC or CODEC settings
    3. Render in black and white (BW button)
    4. If using FFMPEG, do not activate Multiplex audio
    5. If using FFMPEG, Autosplit Output (Video panel button)
    6. Render only RGB if you need color; the A channel (RGBA button) takes more memory and is unused when saving a movie file.

User Manual: Contents | Guidelines | Blender Version **2.43**

Introduction

Compositing refers to building up your image or file from components. Like plywood, the final product is a combination of layers bonded together. The layers work together to create a better product than any single piece. Compositing has two main purposes: *Assembly* and *Enhancement*.

Image or sequence *assembly* is where the final footage is assembled from multiple shots of raw footage. For example, the two−second shot of the actor driving is taped to a three−second shot of the car driving by, resulting in a five−second sequence.

Image *enhancement* changes the appearance of a shot. In film, a shot may be taken during a bright and sunny day, but in post−production, the editor/producer may want the sky to be dark and cloudy to convey a sense of foreboding. The end result is thus a composite of what was originally shot, with effects layered on top and bonded to the original.

Compositing is accomplished two ways in Blender:

- Assembly
    - Using the Video Sequence Editor
    - Using Composition Nodes
- Enhancement
    - Using Composition Nodes

The Video Sequence Editor is used to *assemble* your shots into raw footage. The editor allows you to add a few mixing and transition effects, such as alpha over and wipe, and was the first way Blender enabled compositing. Limited post−processing can be done using the editor. This set of tools has been modularized and greatly expanded by the introduction of Composition Nodes. Composition Nodes are little workers that do something specific to a shot, such as colorizing it or sharpening it; you arrange these nodes and thread them together in assembly−line fashion to composite the images, resulting in processed footage.

To learn how to use the new, wild, and very cool Composition Nodes, click here. To use the Video Sequence Editor, click here.

Previous: Manual/Rendering_From_Command_Line	Contents	*Next:* Manual/Compositing Node Editor

User Manual: Contents | Guidelines | Blender Version **2.43**

This section explains the window in general, and its header menu options. It also tells you how to enable nodes for use within Blender.

More information on **where to use nodes** please refer to the following pages:

- Nodes for Materials (types of material nodes).
- Nodes for Composition (types of composition nodes).

To learn more about **how to handle nodes themselves** in general please refer to the following page:

- Manual/Using Nodes.

Also there is a reference page available explaining the nodes windows and connected functions:

- Reference/Windows/Nodes.

Accessing The Node Editor

Select the Node Editor window.

First let's enter the node editor by changing our window type to Node Editor. As shown in *Select the Node Editor window*, click on the window type icon and select Node Editor from the popup list. Node maps can get quite large, so use or create a big window. The window has a graph−paper style background and a header.

Each scene within your blend file can have multiple Material Node map and ONE Compositing Node map. The Node Editor window shows either map, depending on the selector position.

Hint

You might want to add a new window layout called 6−Nodes (the list is shown on the User Preferences header at the top of your screen) comprised mostly of one big Node Editor window. My layout has the buttons window at the bottom and a text editor window on the side for me to keep notes. If you have a widescreen display (or even a regular one), you might also want to add a 3D view or UV/Image Editor window to the left side of the Node window layout, so you can work with images or your model while you're manipulating nodes. Having the 3D Preview Render panel open on top of an object is quite useful if you're tweaking material nodes.

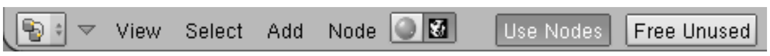
By default, the header, when first displayed, is uninitialized as shown:

Default Node Editor header.

Activating Nodes

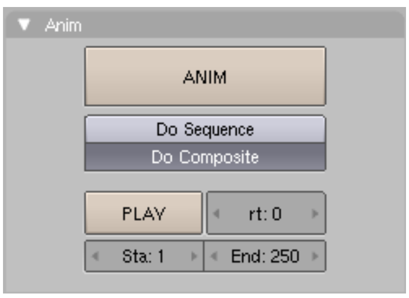
- What nodes to use?
    - If you want to work with a material node map, click the ball in the Material/Compositing node set selector. (see *Node Editor Header with Material Nodes enabled.*)
    - If you want to work with a compositing node map, click the face on the Material/Compositing node set selector. (see *Node Editor Header with Compositing Nodes enabled.*)
- To actually activate nodes, click the Use Nodes button.
- The first time that you select either a Material or a Compsiting node map, the Node Editor window will be instantly filled with starter input and output compositing nodes already connected together.

Node Editor Header with Material Nodes enabled.



Node Editor Header with Compositing Nodes enabled.

Required settings for Composition



Use Composition Nodes.

If you are compositing, you must now tell Blender to use the Node map that has been created, and to composite the image using the Node Map. To do so, click on the Do Composite button located below the Animation button. This tells Blender to composite the final image by running it through the composition node map.

From here, you add and connect nodes in a sort of map layout to your heart's content (or physical memory constraints, whichever comes first). But first, let's lay the groundwork by going over the window in general and the header menu options and buttons.

Node Editor Window Actions

When the cursor is in the window, several standard Blender hotkeys and mouse actions are available, including:

Popup menu
: Space − Brings up a main popup menu, allowing you to add, view, select, etc.

Delete
: X or Del − Deletes the selected node(s).

Box select
: B − Starts the bounding box selection process. Position your cursor and LMB click & drag to select a set of nodes.

Cut connections (box)
: LMB click & drag − Starts a box selection, BUT when you let up the mouse button, all threads (connections) within the box are broken.

Undo
: Ctrl Z Very helpful if you forgot to press B before box−selecting, eh?

Redo
: Ctrl Y or Shift Ctrl Z − You can use this if you used "undo" a bit to often :)

Select multiple
: Shift LMB or Shift RMB − Multiple node select.

Grab/Move
: G − Moves your current selection around.

Execute
: E − pumps inputs through the noodle, refreshing everything.

Standard Window Control

Node maps can get pretty hairy (large and complicated, that is). The contents of the window, (the node map) can be panned just like any other Blender window by clicking MMB and dragging about. Wheeling MW up/down or using the keypad
NumPad +/NumPad − will zoom in/out. The window can be resized and combined using the standard window techniques (see *Navigating in 3d Space*).

Node Editor Header

Node Editor Header with Material Nodes enabled.

Node Editor Header with Compositing Nodes enabled.

On the window header, you will see header options:

- View − to see things more clearly;
- Select − to do things more clearly;
- Add − to walk with...err..to add Nodes, organized by type;
- Node − to do things with selected nodes, akin to vertices;
- a Material or Compositing node set selector;
- a Use Nodes button;
- a Free Unused button.

View, Select, and Add Header Menus

These popup menus provide the basic functions:

View
: This menu changes your view of the window, standing in for the standard keyboard shortcuts NumPad + (zoom in), NumPad − (zoom out), HOME (zoom all) or equivalent mouse actions.

Select
: This menu allows you to select a node or groups of nodes, and does the same as typing the hotkey to select all A or start the border select B process.

Add
: This menu allows you to add nodes. Please see the next section for a discussion on the types of nodes that you can add, and what they do. Clicking this menu item is the same as pressing space when the cursor is in the window

Node Header Menu

Show Cyclic Dependencies
: C − Ok, so you've been adding and connecting nodes to your heart's content, and you haven't run out of memory yet. Selecting Show Cyclic Dependencies will show you where you have connected your threads in a circle. For example, you can easily connect a mix output as input to another node, and then connect that node's output back to the mix node input, resulting in a little circle where the image just runs round and round. Left alone, it will eventually get tired and dizzy and crash your computer.

Hide
H − Hides your selected nodes. Just like vertices in a mesh.

Grouping
Most importantly, this menu option allows you to create a user−defined group of nodes. This group can then be edited and added to the map. To create a group, select the nodes you want, and then Node â†' Make Group, or just use the keyboard shortcut Ctrl G. Edit the name using the little input box in the group. Groups are easily identified by their green header and cool names you have picked for them.

Delete
X − Deletes selected nodes.

Duplicate
Shift D − Makes an Unlinked copy, with the same settings as the original.

Grab
G − Moves the little nodes around according to your mouse, just like with meshes.

Duplicate − Faked you out

The new copy is placed **exactly over the old one**. But it isn't the connected one, so playing with the controls will do nothing to your images, even though it **looks** like it's connected with the little threads coming out of the node that is **underneath**. You have to move the duplicated node to reveal the connected node beneath it.

Grab − Reminder Only

Just like my mother−in−law, the menu item does not actually do anything; it's just there to remind you that you can press the G key when your cursor is in the window and actually accomplish something with your life (like rearranging nodes in the window).

Material/Composting Selector Button

Nodes are grouped into two categories, based on what they operate on. Material Nodes operate on a material in use within the blend file. To work with Material Nodes, click on the ball. When you want to work with Compositing nodes, click on the face to show the Compositing Node map.

Use Nodes Header Button

This button tells the render engine to use the node map in computing the material color or rendering the final image, or not. If not, the map is ignored and the basic render of the material tabs or scene is accomplished.

Free Unused Header Button

This button frees up memory space when you have a very complex node map. Recommended.

Previous: Manual/Compositing	Contents	*Next:* Manual/Using Compositing Nodes

Node Basics

What are nodes

"Nodes" (some people call connected nodes "noodles") are individual blocks that perform a certain operation on zero or more inputs or outputs. Some nodes, like the Value or RGB nodes, only output a value. Other nodes, such as the RGB Curves node, take in an image, and output a modified copy. Other nodes, such as the Defocus Node or Vector Blur node, perform even more complex tasks.

Adding and Arranging Nodes

Nodes are added in two ways to the node editor window:

- By clicking the Add menu in the node editor toolbar and picking the type of node you want, or
- By having your cursor in the node editor window and pressing Space and picking a node from the popup Add menu.

In general, try to arrange your nodes within the window such that the image flows from left to right, top to bottom. Move a node by clicking on a benign area and dragging it around. The node can be clicked almost anywhere and dragged about; connections will reshape as a bezier curve as best as possible.

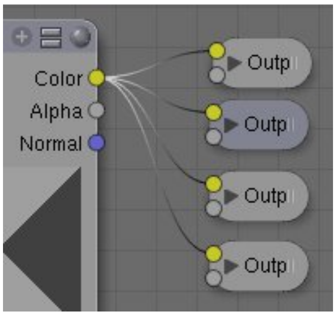
Threads beneath Nodes

Threads (the curves that connect sockets) may reposition behind a node; however they are just that and do not interact with that node in any way.

Sockets

Node Sockets.

Each Node in your node window will have "sockets" (often also referred to as "connectors") which are small colored circles to which input data and output data will be linked (*Node Sockets*).



Node Linking.

There are three colors:

Yellow sockets
: Indicates that color information needs to be input or will be output from the node.

Grey sockets
: Indicates values (numeric) information. It can either be a single numerical value or a so−called "value map" (you can think if a value map as a grayscale−map where the different amount of bright/dark reflects the value for each point.) If a single value is used as an input for a "value map" socket all points of the map are set to this same value.
Common use: Alpha maps and value−options for a node.

Blue/Purple sockets
: Indicates vector/coordinate/normal information.

Between nodes, yellow must be linked to yellow, gray to gray, blue to blue, unless you use a converter which we'll cover later on.

Next to the color in the node you will see the name of that socket. Though not always the case, you can see the name of the socket as what the information is *intended* to be, not necessarily what it *has* to be − for example, I can add a link from an gray socket titled Alpha to the material node's gray Reflection socket and still get a result, they key thing being that it's a gray to gray connection.

There are exceptions where you can mix yellow (e.g. a color−image) and gray (e.g. grayscale) without convertors, Blender normally places a convertor if needed, so feel free to experiment with them. You can use the "Viewer" output nodes as explained in the later sections to see if/how it works.

Connecting and Disconnecting Sockets

You link between sockets by clicking the socket with the LMB and holding to drag the thread to another socket, you then let go once you reach the corresponding socket. To break a link between sockets click the LMB and hold and drag a box around any part (it can be really small) to break the link. From output sockets, multiple threads can be extracted and attached to many nodes (*Node Linking*).

In this case, a copy of each output is routed along a thread. However, only a single thread can be linked to an input socket.

Node Groups

Both material and composite nodes can be grouped. Grouping nodes can simplify the node network layout in

the node editor, making your material or composite 'noodle' (node network) easier to work with. Grouping nodes also creates what are called NodeGroups (inside a .blend file) or NodeTrees (when appending).

If you have created a material using nodes that you would like to use in another .blend file, you can simply append the material from one .blend file to another.

However, what if you would like to create a new material, and use a branch from an existing material node network? You could re−create the branch. Or you could append the material to the new .blend file, then cut and paste the branch that you want into the new material. Both of these options work, but are not very efficient when working across different .blend files.

What if you have created a â€œDepth of Fieldâ€ composite node network and would like to use it in another .blend file? Here again, you could re−create the network, but this is not very efficient.

A better method of re−use for either material node branches or composite node networks would be to create groups of nodes. These groups will then be made available through the Blender's library and standard appending method.

Grouping Nodes

To create a node group, in the node editor, select the nodes you want to include, then press Ctrl G or Space â†’ node â†’ make group. A node group will have a green title bar. All of the selected nodes will now be minimized and contained within the group node. Default naming for the node groups is NodeGroup, NodeGroup.001 etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful. When appending node groups from one .blend file to another, Blender does not make a distinction between material node groups or composite node groups, so I recommend some naming convention that will allow you to easily distinguish between the two types. For example, name your material node branches Mat_XXX, and your composite node networks Cmp_XXX.

What NOT to include in your groups.

> Material node groups should not include material nodes or output nodes. If you include a material node in your group, you will end up having the material node appear twice, once inside the group, and once outside the group in the new material node network. If you include an output node in the group, there will not be an output socket available from the group.
>
> Composite node groups can not include a Render Layer node (Blender wont let you), and should not contain a Composite output node. Here again, if they include any connected Output node (Viewer, Split Viewer, etc) then the Group will not have an image output socket.

Editing Node Groups

With a group node selected, pressing Tab expands the node to a window frame, and the individual nodes within it are shown to you. You can move them around, play with their individual controls, re−thread them internally, etc. just like you can if they were a normal part of your editor window. You will not be able to thread them to an outside node directly from them; you have to use the external sockets on the side of the Group node. To add or remove nodes from the group, you you need to ungroup them.

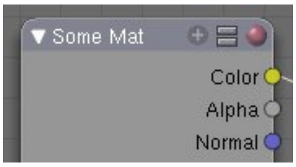
Ungrouping Nodes

The Alt G command destroys the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread internal nodes to other nodes in your workspace.

Appending Node Groups

Once you have appended a NodeTree to your .blend file, you can make use of it in the node editor by pressing Space â†’ Add â†’ Groups, then select the appended group. The "control panel" of the Group is the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

Node Controls



Top of a Node.

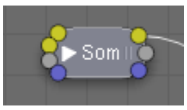
At the top of a node there are up to 4 visual controls for the node (*Top of a Node*). Clicking these controls influences how much information the node shows.

 ***Arrow***
: The arrow on the left collapses the node entirely (*Collapsing Arrow*).

 ***Plus sign (+)***
: The "plus" icon collapses all sockets that do not have a thread connected to it (*Plus Sign*).

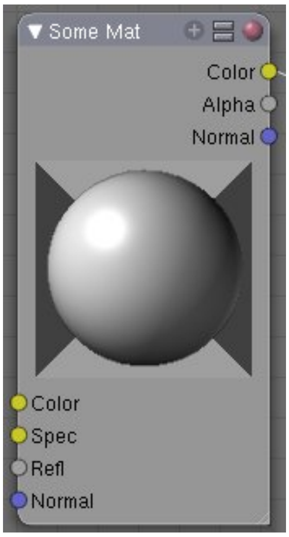
 ***Two squares (=)*** *or "Equal sign"*
: The icon with the two squares collapse all the items in a node that have boxes with information in them (*Menu Collapse*).

 ***Sphere***
: The sphere icon collapses the viewing window (if the node has one) (*Sphere*).
If the Sphere is **red** this can have 3 reasons:
- ◊ It's the only effective output Composite node in the compositor.
- ◊ It's the only effective material Output node (the first one that is added).
- ◊ If it's a Material input node that has a Material (MA:) assigned to it.



Collapsing Arrow.

Plus Sign.

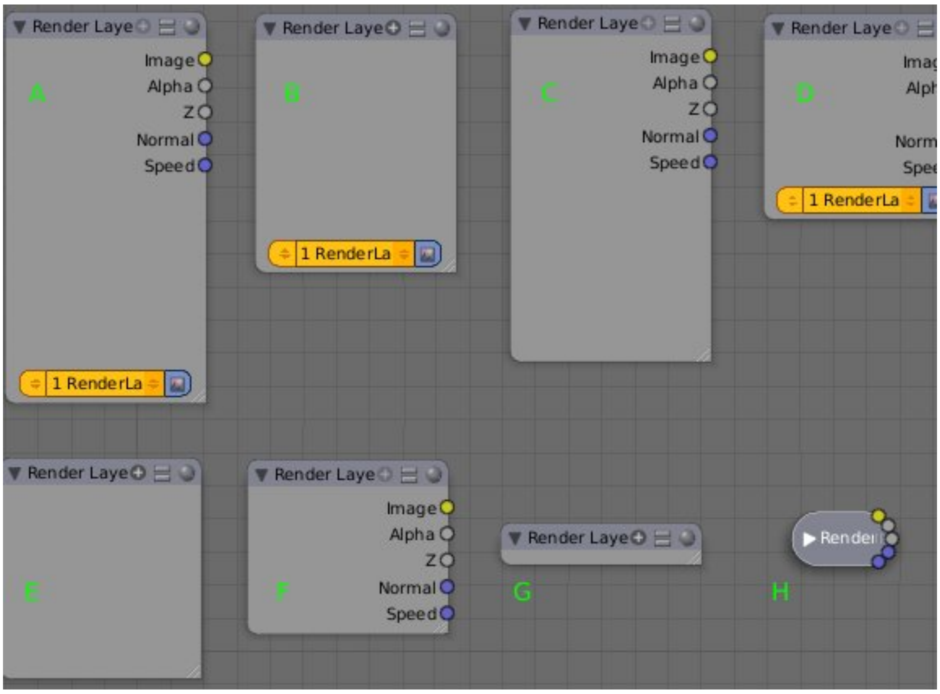


Menu Collapse.

Sphere.

In Combination.

The later three can be used in varying combinations with each other. The arrow that collapses the entire node can only be used in combination with the plus sign (*In Combination*).



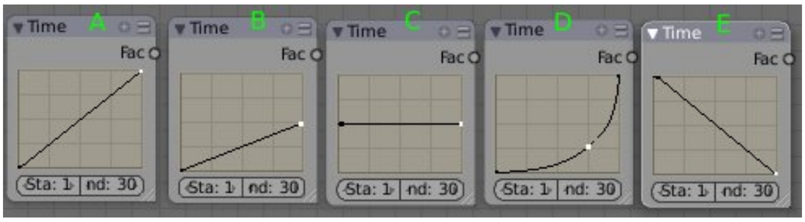
Top sizing controls of a Node
A) Normal, **B**) + Sign clicked, **C**) = Sign clicked, **D**) Sphere clicked, **E**) + and = clicked, **F**) = and Sphere clicked, **G**) All three clicked **H**) Arrow clicked.

Node Sizing

Fine Sizing of an individual node can also be accomplished somewhat by clicking LMB and dragging in the lower right−hand corner (where the little slanted lines are).

Node Curves

Some nodes have a curve area that translates an input value to an output value. You can modify this curve shape by clicking on a control point and moving it, or adding a control point. Some examples are shown below:



Modifying a curve node.

Every curve starts out as a straight line with a slope of 1. (My daughter NEVER thought she would use her high school algebra. Ha!) The curve starts out with two tiny black control points at each end of the line. Clicking LMB on a control point selects it and it turns white.

Changing the curve affects how the output is generated. The input, X, usually proceeds linearly (at regular intervals) across the bottom axis. Go up until you hit the curve, and then over to the right to determine the Y output for that corresponding X. So, for the second example, as X goes from 0 to 1.0 across the bottom, Y varies from 0.0 to 0.5. In the third, as X goes from 0.0 to 1.0 across the bottom, Y stays constant at 0.5. So, in the picture above, these curves have the following affect on time: **A** don't affect, **B** slow down, **C** stop, **D** accelerate, and **E** reverse time.

The "Curves" widget is a built−in feature in Blender's UI, and can be used anywhere, provided the curve data itself is being delivered to this widget. Currently it is in use in the Node Editor and in the UV Window.

This widget will map an input value horizontally and return the new value as indicated by the height of the curve.

RGB Curves

Multiple curves can be edited in a single widget. The typical use, RGB curves, has "Combined" result or "Color" ("C") as the first curve, and provides curves for the individual R, G, and B components. All four curves are active together, the "C" curve gets evaluated first.

Selecting curve points

- LMB always selects 1 point and deselects the rest.
- Hold Shift while clicking to extend the selection or select fewer points.

Editing curves

- LMB click&drag on a point will move points.
- A LMB click on a curve will add a new point.
- Dragging a point exactly on top of another will merge them.
- Holding Shift while dragging snaps to grid units.
- Ctrl LMB adds a point.
- Use the X icon to remove selected points.

Editing the view

The default view is locked to a 0.0−1.0 area. If clipping is set, which is default, you cannot zoom out or drag the view. Disable clipping with the icon resembling a #.

- LMB click&drag outside of curve moves the view
- Use the + and − icons to zoom in or out.

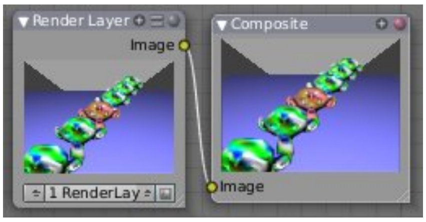
Special tools

The wrench icon gives a menu with choices to reset a view, to define interpolation of points, or to reset the curve.

Previous: Manual/Compositing Node Editor	Contents	*Next:* Manual/Node Composition

User Manual: Contents | Guidelines | Blender Version **2.43**

Compositing Nodes allow you assemble and enhance an image (or movie) at the same time. Using composition nodes, you can glue two pieces of footage together and colorize the whole sequence all at once. You can enhance the colors of a single image or an entire movie clip in a static manner or in a dynamic way that changes over time (as the clip progresses). In this way, you use composition nodes to both assemble video clips together, and enhance them.



To process your image, you will use nodes to import the image into Blender, change it, merge it with other images, and finally save it. Think of nodes as workers on an assembly line. Each worker performs one basically simple task before passing the product on to the next worker. The assembly line is called a thread, and the image enters and leaves a node by sockets. You connect one node to another by creating a thread that runs from one node's output socket to another node's input socket. A bunch of connected nodes is called a **noodle**. The example to the right shows the simplest noodle; an input node threads the camera view to an output node so it can be saved.

Compositing nodes are connected by threads into a map that routes an image through a sequence of nodes. Each node performs some operation of the image. You create the map using the Node Editor window type. Before you continue reading, it is elementary that you understand how to use this window type. Read all about the Node Editor by clicking here.

Term: Image

We use the term *Image* to refer to a single picture, a numbered sequence of images that, when played flipbook style, result in a movie clip, a single frame of a movie sequence, or a movie clip.

Every node has a similar set of controls and conventions. For more information on working with node controls, click here.

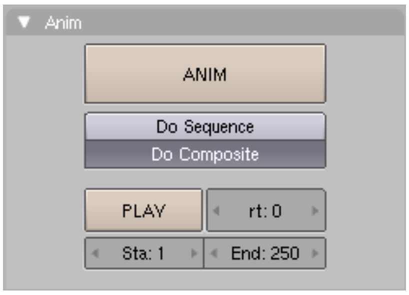
Accessing and Activating Nodes

Select the Node Editor window

Access the [Node Editor] and enable Composite Nodes by clicking on the correct icon (picturing a Face).

Node Editor Header with Composite Nodes enabled.

To activate nodes for the composition, click the Use Nodes button. Blender creates a default starting noodle, consisting of two nodes threaded together.



Use Composition Nodes

To use this mini−map, you must now tell Blender to use the Compositing Node map that has been created, and to composite the image using composition nodes. To do so, switch to the Scene button area (F10) and activate the **Do Composite** button located below the Animation button. This tells Blender to composite the final image by running it through the composition node map.

Working with the Composition Map

You now have your first noodle, a RenderLayer input node threaded to an Composite output node. From here, you add and connect nodes in a sort of map layout to your heart's content (or physical memory constraints, whichever comes first). There are many types of compositing nodes that do many different things. The available nodes are discussed in the next section, which you can quickly access [by clicking here].

As your map grows, you will need to keep it organized. Try to arrange the 'assembly line' of nodes from left to right, top to bottom. You can also group nodes together into subroutines, as discussed below.

Grouping Composition Nodes

As your map grows, you might find it helpful to organize your nodes by grouping some of them together. Blender allows you to make user−defined groups of nodes.

Panel: Node Editor ***Node Composition***

Menu: Shift A â†’ Groups

Node maps can get quite complex. Blender allows you to group a set of nodes together to both save space and help you conceptualize what the net effect of a mini−map does to a material/image. This menu selection shows the names of the groups of nodes that you have defined. Select any one to add that group to the map.

Create a Node Group

A group is created by Shift−clicking all the nodes you want in the group, and then selecting Node −> Make Group (Ctrl G). The group node is shown with a green bar, and the name is shown in an editable field (Shift−click on the name to enter EditMode and change the name to something that represents what that group does. The input sockets to the group is the input sockets of contained nodes, and akin for output(s).

Editing Node Groups

You can select the group and press Tab (or select Node −> Edit Group) to enter/leave the group.

Ungroup Nodes

You can select the group and press Alt G (or select Node −> Ungroup) to convert the grouped nodes back to normal.

A "FadeToBlack" compositing group would be a Time node that feeds the Factor of an RBG Curve node. The curve in the RGB Curve node would be a flat line at zero. Namely, any X that came in results in a zero output (black). The factor starts at zero (no effect on the input image), and cranks up to one (full effect, namely black). The input sequence will fade to black over the frame count set in the timer.

Some Examples of Using Compositing Nodes

You can do just about anything with images using nodes. The "Fade to Black" noodle group above is a simple example of what you can do using compositing nodes.

Raw footage from a foreground actor in front of a blue screen, or a rendered object doing something can be layered on top of a background. Composite both together, and you have composited footage.

You can change the mood of an image:

- To make an image 'feel' colder, a blue tinge is added to the pic.

- To convey a flashback or memory, the image may be softened.
- To convey hatred and frustration, add a red tinge or enhance the red. The film 'Sin City' is the most extreme example of this I have ever seen.
- A startling event may be sharpened and contrast enhanced.
- A happy feeling − you guessed it − yellow (equal parts red and green no blue) for bright and sunny.
- Dust and airborne dirt is often added as a cloud texture over the image to give a little more realism.

Terms

Compositing works equally well on a static image or a video sequence. We use the term **image** here to represent either. A **Noodle** is a set of connected nodes.

Previous: Manual/Using Compositing Nodes	Contents	*Next:* Manual/Compositing Nodes

User Manual: Contents | Guidelines | Blender Version **2.43**

The compositing nodes allow you to manipulate an image by routing it through a map of connected nodes. A starting image is routed through different nodes that do various things to the image, combined with other inputs or put back together, and finally output. Images can be split into their RGB components, combined (mixed) with other inputs, and layered on top of one another. In film, this process is called post−processing and occurs during the post−production (namely, after the raw shots are 'in the can'). At this phase, the raw footage is handed over to an editor who does all sorts of magic to make a good looking movie. Now you can too!

This section is organized by type of nodes, which are grouped based on similar functions:

- Input − Adds something to the node map, such as an image or a value.
- Output − Displays the result in progress as a small image.
- Color − Manipulates the colors of an image.
- Vector − Manipulate the intensities and reflections of an image
- Filters − Process the image to enhance it, working on adjacent pixels.
- Convertors − Separate the image into its component video, or convert formats.
- Mattes − Generating mattes to mask off areas of an image.
- Distortion − Changing the shape of the image.
- Groups − User−defined groups of nodes.

The simplest way to add a node is to put your cursor in a Node Editor window, and then press Space and click on Add. The popup menu will extend to show you these types of nodes. Click on a node type and the popup menu will extend again to show you specific node types available.

Previous: Manual/Node Composition	Contents	*Next:* Manual/Compositing Nodes Input

User Manual: Contents | Guidelines | Blender Version **2.43**

A starting image can be:

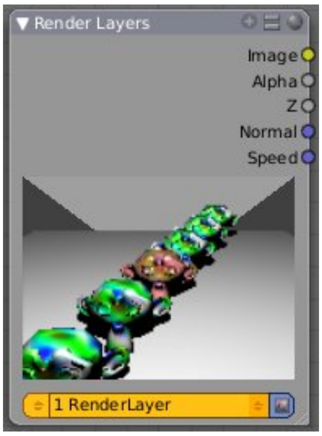
- taken directly from the active camera in a selected scene,
- from a JPG, PNG, etc. file as a static picture,
- a movie clip (such as an animation sequence or home movie), or
- just a color.

In addition, these nodes generate information that feeds other nodes.

Render Layers Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Input â†’ Render Layers



Render Layers Node

This node is the starting place to getting a picture of your scene into the compositing node map.

This node inputs an image from a scene within your blend file. Select the scene, and the active render layer from the yellow selection list at the bottom of the node. Blender uses the active camera for that scene to create an image of the objects specified in the RenderLayer.

The Image is input into the map, along with the following data:

- Alpha (transparency) mask

Depending on the Renderlayer passes that are enabled, other sockets are available. By default the Z is enabled:

- Z depth map (how far away each pixel is from the camera)

The example shows that two other passes are enabled:

- Normal vector set (how light bounces off the surface)
- Speed vector set (how fast an object is moving from one frame to the next)

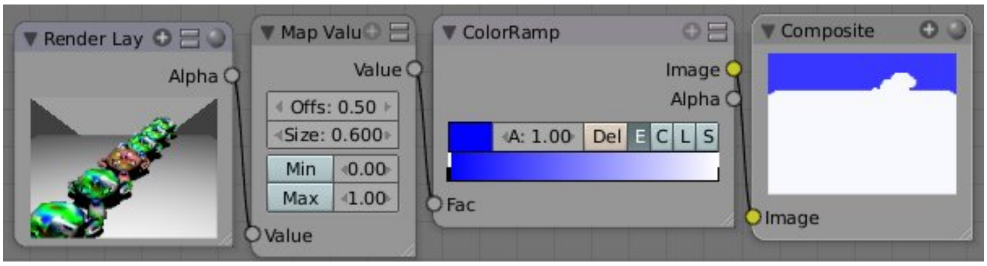
Use the re−render button (Small landscape icon − to the right of the Renderlayer name) to re−render the scene and refresh the image and map.

You may recall that a .blend file may contain many scenes. The Renderlayer node can pick up the scene info from any available scene by selecting the scene from the left−hand selector. If that *other* scene also uses the compositor and/or sequencer, you should note that the scene information taken is the raw information (pre−compositing and pre−sequencing). If you wish to use composited information from another scene, you will have to render that scene to a multilayer OpenEXR frameset as an intermediate file store, and then use the Image input node instead.

Using the Alpha Socket

Using the Alpha output socket is crucial in overlaying images on top of one another and letting a background image "show through" the image in front of it.

In a Blender scene, your objects are floating out there in virtual space. While some objects are in front of one another (Z depth), there is no ultimate background. Your world settings can give you the illusion of a horizon, but it's just that; an illusion. Further, some objects are semi−transparent; this is called having an Alpha value. A semi−transparent object allows light (and any background image) to pass through it to the camera. When you render an image, Blender puts out, in addition to a pretty image, a map of what solid objects actually are there, and where infinity is, and a map of the alpha values for semi−transparent objects. You can see this map by mapping it to a blue screen:



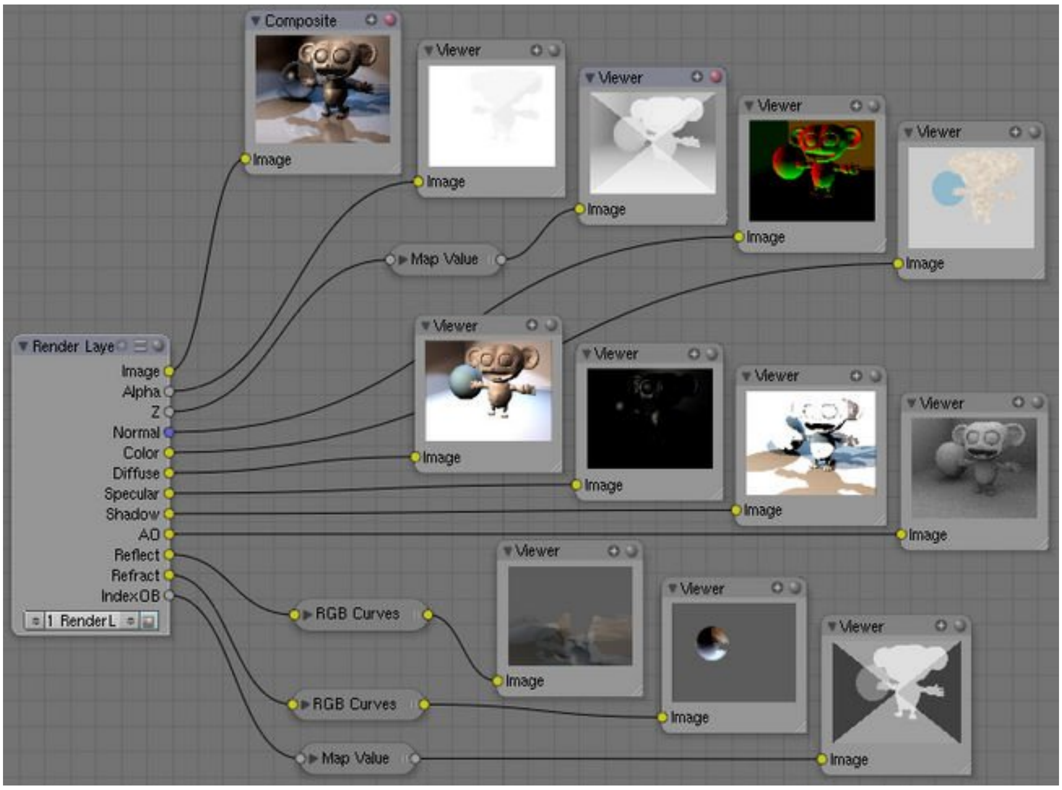
Viewing the Alpha values

In the little node map above, we have connected the Alpha output socket of the RenderLayer node to a Map Value node (explained later, but basically this node takes a set of values and maps them to something we can use). The Color Ramp node (also explained later in detail) takes each value and maps it to a color that we can see with our eyes. Finally, the output of the Color Ramp is output to a Composite viewer to show you, our dear reader, a picture of the Alpha values. Notice that we have set up the map so that things that are perfectly solid (opaque) are white, and things that are perfectly transparent (or where there is nothing) are blue.

Optional Sockets

For any of the optional sockets to appear on the node, you MUST have the corresponding pass enabled. In order for the output socket on the RenderLayer node to show, that pass must be enabled in the RenderLayer panel in the Buttons window. For example, in order to be able to have the Shadow socket show up on the RenderLayer input node, you must have the "Shad" button enabled in the Buttons window, Scene Render buttons, Renderlayer panel. See the RenderLayer tab (Buttons window, Output frame, Render Layers tab, Passes selector buttons) for Blender to put out the values corresponding to the socket.

For a simple scene, a monkey and her bouncy ball, the following picture expertly provides a great example of what each pass looks like:



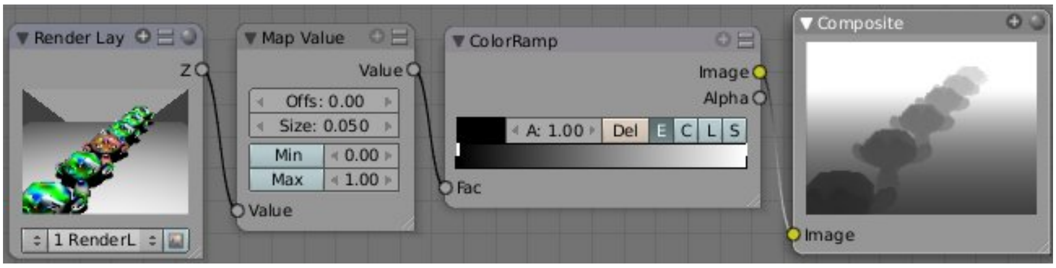
The available sockets are:

- Z: distance away from the camera, in Blender Units
- Normal (Nor): How the color is affected by light coming from the side
- UV: how the image is distorted by the UV mapping
- Speed (Vec): How fast the object is moving, and it what direction
- Color (Col): the RGB values that color the image that you see.
- Diffuse: the softening of colors as they diffuse through the materials
- Specular: the degree of shinyness added to colors as they shine in the light
- Shadow: shadows cast by objects onto other objects
- AO: how the colors are affected by Ambient Occlusion in the world
- Reflect (Ref): for mirror type objects, the colors they reflect and are thus not part of their basic material
- Refract: how colors are bent by passing through transparent objects
- Radio (Radiosity): colors that are emitted by other objects and cast onto the scene
- IndexOB: a numeric ordinal (index) of each object in the scene, as seen by the camera.

Using the Z value Socket

Using the Z output socket is crucial in producing realistic images, since items farther away are blurrier (but more on that later).

Imagine a camera hovering over an X−Y plane. When looking through the camera at the plane, Y is up/down and X is left/right, just like when you are looking at a graph. The camera is up in the air though, so it has a Z value from the X−Y plane, and, from the perspective of the camera, the plane, in fact all the objects that the camera can see, have a Z value as a distance that they are away from it. In addition to the pretty colors of an image, a RenderLayer input node also generates a Z value map. This map is whole bunch of numbers that specify how far away each pixel in the image is away from the camera. You can see this map by translating it into colors, or shades of gray:



Viewing the Z values

In the little node map above, we have connected the Z output socket of the RenderLayer node to a Map Value node (explained later). This node takes a set of values and maps them to something we can use. The Color Ramp node (also explained later in detail) takes each value and maps it to a shade of gray that we can see with our eyes. Finally, the output of the colorramp is output to a Composite viewer to show you, our dear reader, a picture of the Z values. Notice that we have set up the Map Value node so that things closer to the camera appear blacker (think: black is 0, less Z means a smaller number) and pixels/items farther away have an increasing Z distance and therefore get whiter. We chose a Size value of 0.05 to see Z−values ranging from 0 to 20 (20 is 1/0.05).

Using the Speed Socket

Even though things may be animated in our scene, a single image or frame from the animation does not portray any motion; the image from the frame is simply where things are at that particular time. However, from the Render Layers node, Blender puts out a vector set that says how particular pixels are moving, or will move, to the next frame. You use this socket to create a blurring effect. Find out more by clicking here.

Image node

Panel: Node Editor Node Composition

Menu: Shift A â†' Input â†' Image

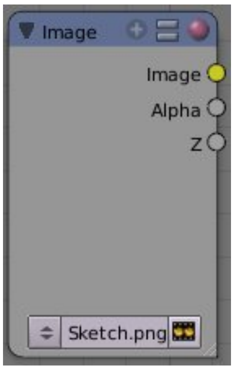


Image node

The Image node injects any image format that is supported by Blender. Besides inputting the actual image, this node can also input Alpha and depth (Z) values if the image has them. If the image is a MultiLayer format, all saved render passes are input. Use this node to input:

- A single image from a file (such as a JPG picture)
- Part or all of an animation sequence (such as the 30th to 60th frame)

- Part or all of a movie clip (such as an AVI file)
- the image that is currently in the UV/Image Editor (and possibly being painted)
- an image that was loaded in the UV/Image Editor

Animated image sequences or video files can also be used. See Animations below.

To select an image file or generated image from the UV/Image Editor, click on the small arrow selector button to the left of the name and pick an existing image (e.g. loaded in the UV editor or elsewhere) or click on LOAD NEW to select a file from your hard disk via a file−browser. These images can be e.g. previously rendered images, matte paintings, a picture of your cat, whatever. Blender really doesn't care.

If the image is part of a sequence, manually click the Image Type selector to the right of the name, and select *Sequence*. Additional controls will allow you to define how much of the sequence to pull in (see Animations below). If the file is a video file, these controls will automatically appear.

Image Channels

When the image is loaded, the available channels will be shown as sockets on the node. As a minimum, the Image, Alpha, and Z channels are made available. The picture may or may not have an alpha (transparency) and/or Z (depth) channel, depending on the format. If the image format does not support A and/or Z, default values are supplied (1.0 for A, 0.0 for Z).

- Alpha/Transparency Channel
  - If a transparency channel is detected, the Alpha output socket will supply it.
  - If it does not have an Alpha channel (e.g. JPG images), Blender will supply one, setting the whole image to completely opaque. (An Alpha of 1.00, which will show in a Viewer node as white − if connected to the Image input socket).
- Z/depth Channel
  - If a Z (depth) channel is detected, the Z output socket will supply it.
  - If it does not have a Z channel (e.g. JPG or PNG images), Blender will supply one, setting the whole image to be at camera (a depth of 0.00). To view the Z−depth channel, use the Map Value to ColorRamp noodle given above in the Render Layer input node, in the Render Layer input node.

Formats

Blender supports many image formats. Currently only the OpenEXR image format stores RGB (color), A (alpha), and Z (depth) buffer information in a single file, if enabled.

Saving/Retrieving Render Passess

Blender can save the individual Render Layers and specific passes in a MultiLayer file format, which is an extension of the OpenEXR format. In this example, we are reading in frames 50 to 100 of 1 RenderLayer that were generated some time ago. The passes that were saved were the Image, Alpha, Z, Specular and AO pass.

To create a MultiLayer image set, simply disable Do Composite, set your Format to MultiLayer, enable the Render Layer passes you wish to save over the desired frame range, and Animate. Then, in Blender, enable Compositing Nodes and Do Composite, and use the Image input node to read in the EXR file. When you do, you will see each of the saved passes available as sockets for you to use in your compositing noodle.

Image Size

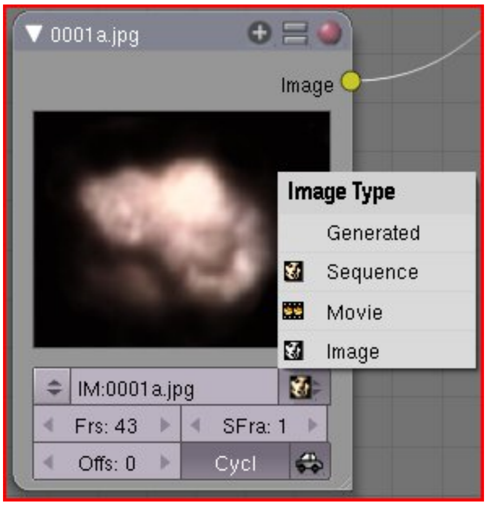
Size matters − Pay attention to image resolution and color depth when mixing and matching images. Aliasing (rough edges), color *flatness*, distorted images, can all be traced to mixing inappropriate resolutions and color depths.

The compositor can mix images with any size, and will only perform operations on pixels where images have an overlap. When nodes receive inputs with differently sized Images, these rules apply:

- The first/top Image input socket defines the output size.
- The composite is centered by default, unless a translation has been assigned to a buffer with the Translate node.

So each node in a composite can operate on different sized images, as defined by its inputs. Only the Composite output node has a fixed size, as defined by the Scene buttons (Format Panel − F10). The Viewer node always shows the size from its input, but when not linked (or linked to a value) it shows a small 320x256 pixel image.

Animations



To use image sequences or movies within your composition, press the face or little film strip button located to the right of the selector. As you click, a pop−up will offer you four choices:

1. Generated −
2. Sequence − a sequence of frames, each frame in a separate file.
3. Movie − a sequence of frames packed into a single .avi or .mov file
4. Image − a single frame or still image in a file

A Movie or Image can be named anything, but a Sequence must have a digit sequence somewhere in its filename, for example fire0001set.jpg, fire0002set.jpg, fire0003set.jpg and so on. The number indicates the frame.

If a Sequence or Movie is selected, an additional set of controls will appear that allows you to select part or all of the sequence. Use these controls to specify which frames, out of the original sequence, that you want to introduce into the animation you are about to render. You can start at the beginning and only use the beginning, or even pick out a set of frames from the middle of an existing animation.

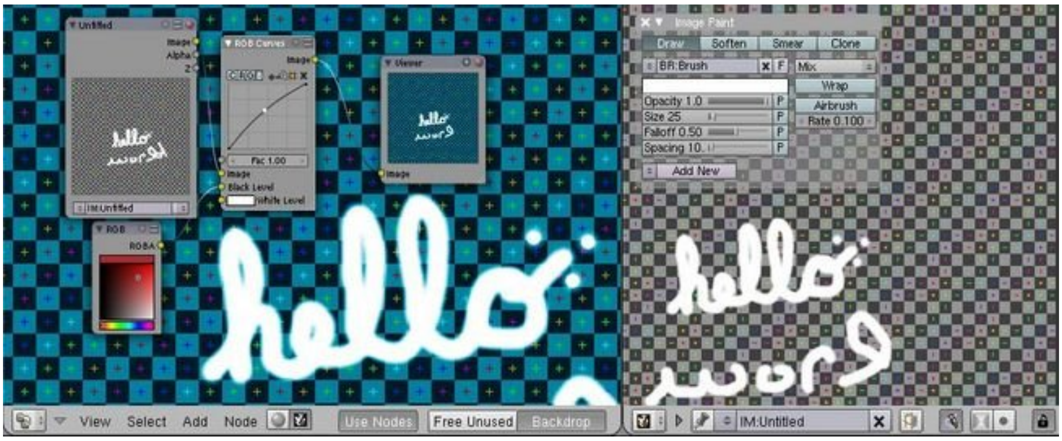
The Frs number button is the number of frames in the sequence that you want to show. For example, if you want to show 2 seconds of the animation, and are running 30 fps, you would put 60 here.

The SFra number button sets the start frame of the animation; namely, at what point in the animation that you *are going to render* do you want this sequence to start playing. For example, if you want to introduce this clip ten seconds into the composite output, you would put 300 here (at 30 fps).

The First number button sets the first number in the animated sequence name. For example, if your images were called "credits−0001.png", "credits−0002.png" through "credits−0300.png" and you wanted to start picking up with frame 20, you'd put 20 here.

To have the movie/sequence start over and repeat when it is done, press the Cyclic button. For example, if you were compositing a fan into a room, and the fan animation lasted 30 frames, the animation would start over at frame 31, 61, 91, and so on, continuously looping. As you scrub from frame to frame, to see the actual video frame used for the current frame of animation, press the auto button to the right of the Cyclic button.

Generated Images



Using the Nodes to modify a painting in progress in the UV/Image window

Blender features [Texture Paint] which works in the UV/Image Editor, that allows you to paint on the fly, and the image is kept in memory or saved. If sync lock is enabled (the lock icon in the header), changes are broadcast throughout Blender as soon as you lift the mouse button. One of the places that the image can go is to the Image Input node. The example shows a painting session going on in the right−hand UV/Image Editor window for the painting "Untitled". Create this image via Image−>New in the UV/Image Editor. Refer to the texture paint section of the user maual for more info on using Texture Paint.

In the left−hand window, the Image input node was used to select that "Untitled" image. Notice that the Image type icon is blank, indicating that it is pulling in a Generated image. That image is colorized by the noodle, with the result used as a backdrop in the Node Editor Window.

Using this setup and the Generated Image type is like painting and post−processing as you continue painting. Changes to either the painting or the post−pro noodle are dynamic and real−time.

Notes

No Frame Stretching or Compression: If the input animation (avi or frame set) was encoded at a frame rate that is *different* from your current settings, the resultant animation will appear to run faster or slower. Blender Nodes do not adjust input video frame rates. Use the scale control inside the [Video Sequence Editor] to stretch or compress video to the desired speed, and input it here. You can incorporate "Slow−Mo" into your video. To do so, ANIMate a video segment at 60 frames per second, and input it via this node, using Render settings that have an animation frame rate of the normal 30 fps; the resulting video will be played at half speed. Do the opposite to mimic Flash running around at hyperspeed.

AVI (Audio Video Interlaced) files are encoded and often compressed using a routine called a *Codec*. You must have a codec installed on your machine and available to Blender that understands and is able to read the file, in order for Blender to be able to de−code and extract frames from the file. If you get the error message **FFMPEG or unsupported video format** when trying to load the file, you need to get a Codec that understands the video file. Contact the author of the file and find out how it was encoded. An outside package, such as VirtualDub, might help you track this information down. Codecs are supplied by video device manufacturers, Microsoft, DivX, and Xvid, among others, and can often be downloaded from their web sites for free.

Splicing Video Sequences using Nodes

The above animation controls, coupled with a little mixing, is all you need to splice video sequences together. There are many kinds of splices:

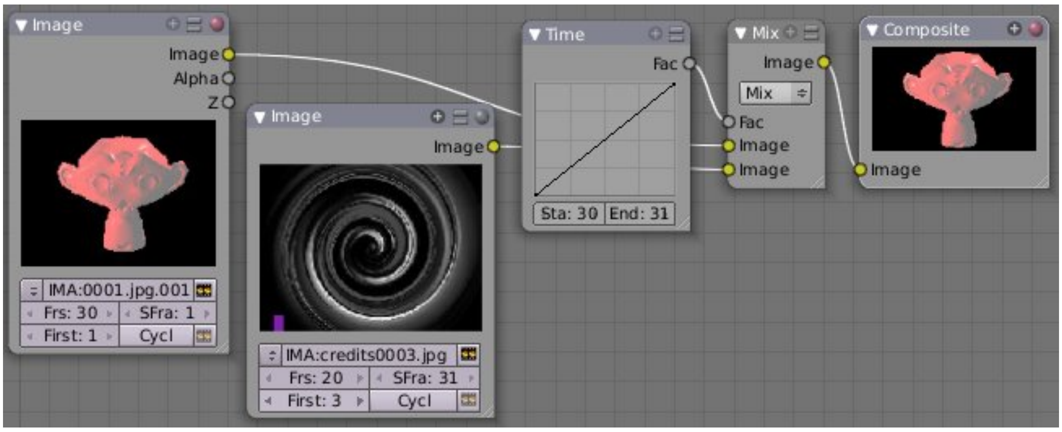
- Cut Splice − literally the ends of the footage are just stuck together
- Fade In − The scene fades in, usually from black
- Fade Out − The scene fades out, usually to black
- Mix − Toward the end of one scene, the images from the next scene meld in as the first scene fades
- Winking and Blinking − fading one cut out while the other fades in, partially or totally through black
- Bumps and Wipes − one cut bumps the other one out of frame, or wipes over it (like from the top left corner down)

Cut Splicing using Nodes

In the example noodle below, we have two pieces of footage that we want to cut splice together.

- Magic Monkey − named 0001.png through 0030.png
- Credits − named credits0001.png through credits0030.png

The editor has reviewed the Credits and thought the first two frames could be thrown away (onto the cutting room floor, as they say) along with the last 8, leaving 20 frames from the total shot. Not shown in this image, but crucial, is that in the Output panel, we set our render output filename to "Monkey−Credits−", and our Animation start and end frames to 1 and 50 (30 from the Monkey, 20 from the credits). Notice the Time node; it tells the Mix node to use the top image until frame 30, and then, at frame 31, changes the Mix factor to 1, which means to use the bottom set of images.



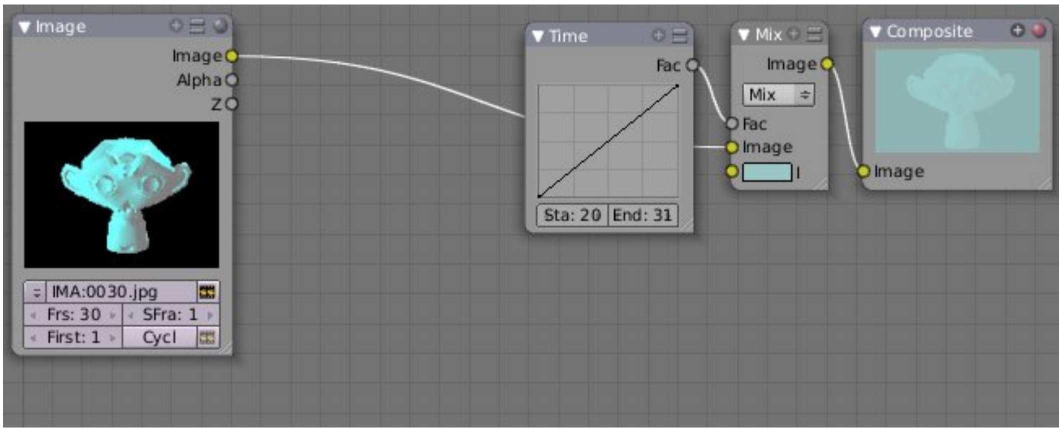
Cut Splice using Nodes

Upon pressing the ANIM button, Blender will composite the animation. If you specified an image format for output, for example, PNG, Blender will create 50 files, named "Monkey−Credits−0001.png" through "Monkey−Credits−0050.png". If you specified a movie format as output, such as AVI−JPEG, then Blender will create only one file, "Monkey−Credits−.avi", containing all 50 frames.

Use cut scenes for rapid−fire transition, conveying a sense of energy and excitement, and to pack in a lot of action in a short time. Try to avoid cutting from a dark scene to a light one, because it's hard on the eyes. It is very emotionally contrasting, and sometimes humorous and ironic, to cut from a very active actor in one scene to a very still actor in another scene, a la old Road Runner and Coyote scenes.

Fade Splicing using Nodes

In the previous topic, we saw how to cut from one sequence to another. To fade in or out, we simply replace one set of images with a flat color, and expand the Time frame for the splice. In the image below, beginning at frame 20, we start fading **out** to cyan:



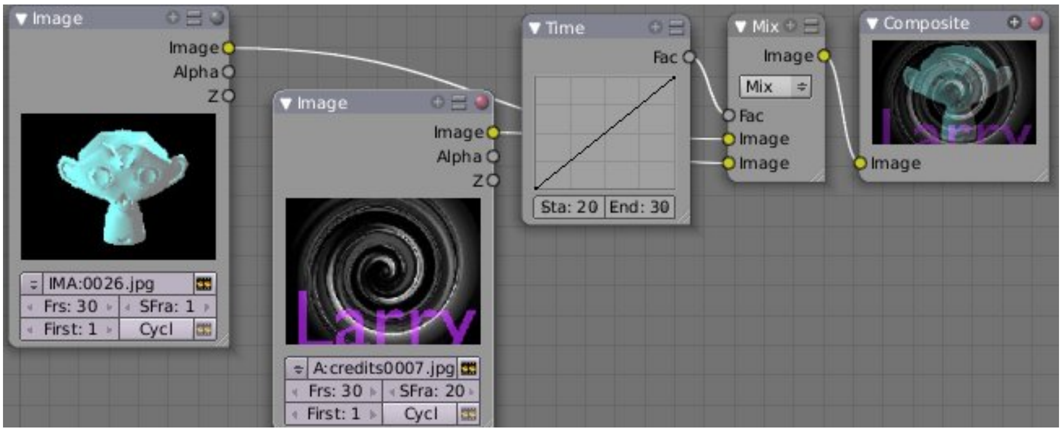
Fading Out using Nodes

Cyan was chosen because that is the color of the Monkey at that time, but you can just as easily choose any color. The image below shows frame 30, when we have almost faded completely.

To fade **in**, change the Mix node and plug the image sequence into the bottom socket, and specify a flat color for the top socket.

Mix Splice using Nodes

To mix, or crossover, from one scene to the next, start feeding the second scene in while the first is mixing out. The noodle below shows frame 25 of a mix crossover special effect to transition from one scene to the next, beginning at frame 20 with the transition completed by frame 30. Action continues in the first scene as it fades out and is mixed with action that starts in the second scene.

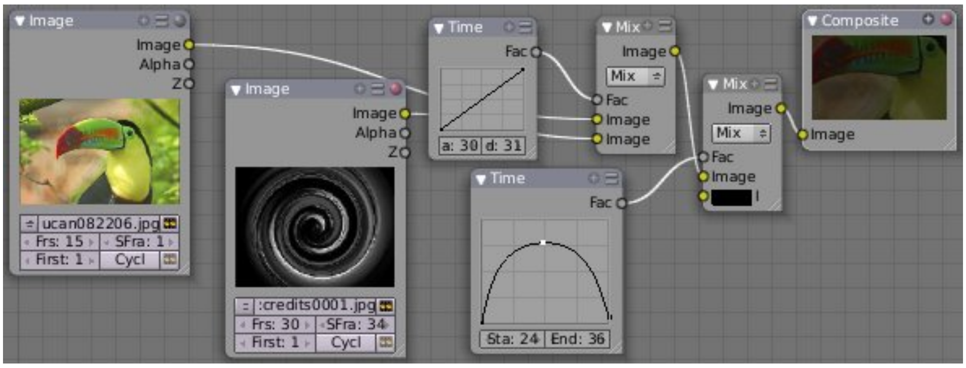


Mix Splice using Nodes

Use this effect to convey similarities between the two scenes. For example, Scene 1 is the robber walking down the street, ending with the camera focusing in on his feet. Scene 2 is a cop walking down the street after him, starting with his feet and working its way up to reveal that the cop is following the robber.

Wink Splice using Nodes

A Wink is just like blinking your eyes; one scene fades to black and the other fades in. To use Blender to get this effect, build on the Cut and Fade splices discussed above to yield:



A Wink using Nodes

In the above example, showing frame 27, we have adjusted some parameters to show you the power of Blender and how to use its Nodes to achieve just the blended crossover effect you desire:

- Postfeed: Even though there were only 15 frames of animation in the Toucan strip, the cutover (top Time node) does not occur until frame 30. Blender continues to put out the last frame of an animation, *automatically extending it for you*, for frames out of the strip's range.
- Prefeed: Even though the swirl does not start playing until frame 34, Blender supplies the first frame of it for Frames 31 through 33. In fact, it supplies this image all the way back to frame 1.
- Partial Fade: Notice the second 'wink' Time node. Like a real wink, it does not totally fade to black; only about 75%. When transitioning between scenes where you want some visual carryover, use this effect because there is not a break in perceptual sequence.

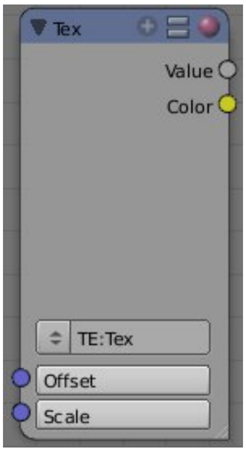
Multiple Feeds

The above examples call out two feeds, but by replicating the Input, Time and Mix nodes, you can have multiple feeds at any one time; just set the Time node to tell the Mixer when to cut over to using it.

Texture Node

Panel: [Node Editor](#) [Node Composition](#)

Menu: Shift A â†’ [Input](#) â†’ Texture



Texture node

The Texture node makes 3D textures available to the compositor. A texture, from the list of textures available in the current blend file, is selected and introduced through the value and/or color socket.

Library

Please read up on the Blender Library system for help on importing and linking to textures in other blender files

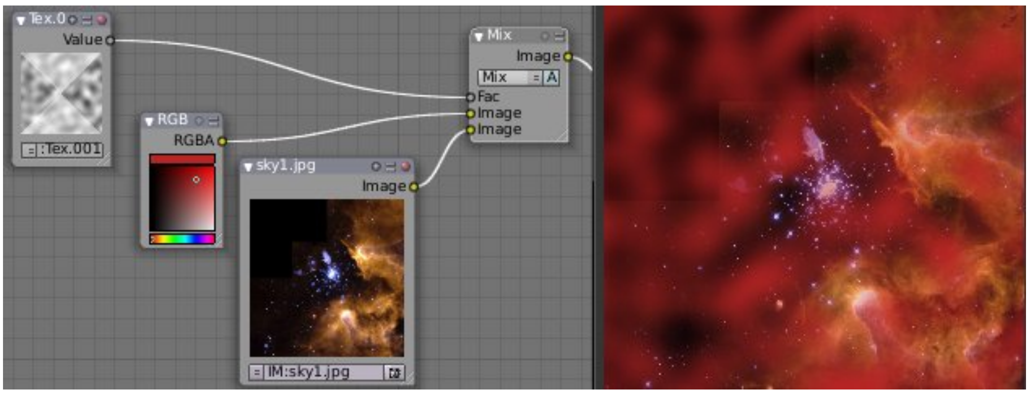
Note

You cannot edit the textures themselves in the node window. To use this node, create and edit the texture in the normal texture buttons, then select the texture from the menu button on the node.

You can change the Offset and a Scale (which is called Offs XYZ and Size XYZ in the Materials Texture Map Input panel) for the texture by clicking on the label and setting the sliders, thus affecting how the texture is applied to the image. For animation, note that this is a vector input socket, because the XYZ values are needed.

Texture nodes can output a straight black−and−white Value image (don't mistake this for alpha) and an image (Color).

Example



In the example above, we want to simulate some red plasma gas out there in space. So, we fog up a image taken from the Hubble telecscope of Orion and take the ever−so−useful Cloud texture and use it to mix in red

with the image.

Value node

Panel: [Node Editor](#) [Node Composition](#)

Menu: Shift A â†’ Input â†’ Value

Value Node

The Value node has no inputs; it just outputs a numerical value (floating point spanning 0.00 to 1.00) currently entered in the NumButton displayed in its controls selection.

Use this node to supply a constant, fixed value to other nodes' value or factor input sockets.

RGB node

Panel: Node Editor Node Composition

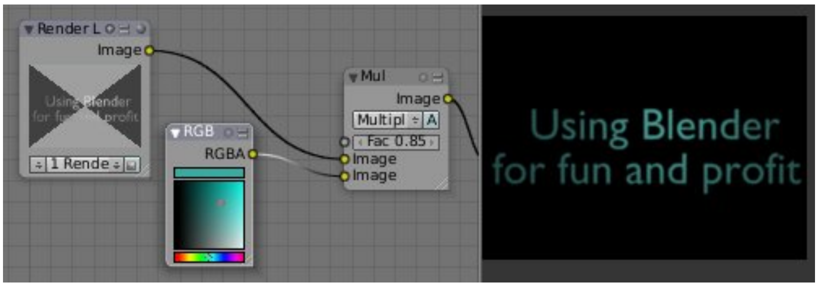
Menu: Shift A â†’ Input â†’ RGB

RGB Node

The RGB node has no inputs. It just outputs the Color currently selected in its controls section; a sample of it is shown in the top box. In the example to the right, a gray color with a tinge of red is slected.

To change the brightness and saturation of the color, LMB click anywhere within the square gradient. The current saturation is shown as a little circle within the gradient. To change the color itself, click anwhere along the rainbow Color Ramp.

Example

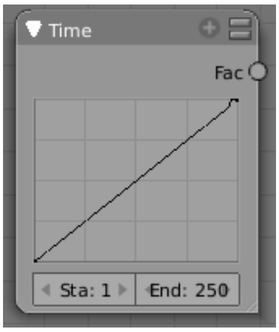


In this example, our corporate color is teal, but the bozo who made the presentation forgot. So, we multiply his lame black and white image with our corporate color to save him from embarassment in front of the boss when he gives his boring presentation.

Time node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Input â†’ Time



Time node

The Time node generates a factor value (from 0.00 to 1.00) (that changes according to the curve drawn) as time progresses through your movie (frames).

The Start and End NumButtons specify the range of time the values should be output along, and this range becomes the X−axis of the graph. The curve defines the Y−value and hence the factor that is output. In the example to the right, since the timespan is 250 frames and the line is straight from corner to corner, 0.50 would be output at frame 125, and 0.75 will be output at frame 187.

Note on output values

The Map Value node can be used to map the output to a more appropriate value. With some time curves, it is possible that the Time node may output a number larger than one or less than zero. To be save, use the Min/Max clamping function of the Map Value node to limit output.

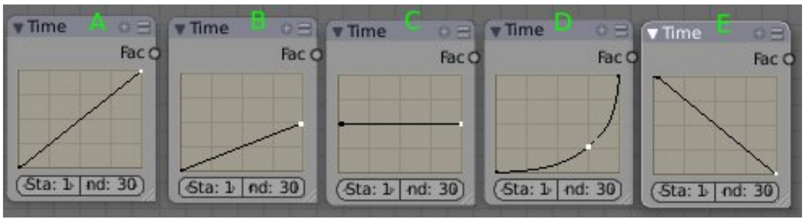
You can reverse time (unfortunately, only in Blender and not in the real world) by specifying a Start frame greater than the End frame. The net effect of doing so is to flip the curve around. Warning: doing so is easily overlooked in your node map and can be very confusing (like meeting your mother when she was/is your age in "Back to the Future").

Time is Relative

In Blender, time is measured in frames. The actual duration of a time span depends on how fast those frames whiz by (frame rate). You set the frame rate in your animation settings (Scene Context F10). Common settings range from 5 seconds per frame for slideshows (0.2 fps), to 30 fps for US movies.

Time Node Examples

In the picture below, over the course of a second of time (30 frames), the following time controls are made:



A) No Effect **B**) Slow Down **C**) Freeze **D**) Accelerate **E**) Reverse

Common uses for this include a "fade to black", wherein the accelerate time curve (typically exponentially−shaped) feeds a mix value that mixes a constant black color in, so that the blackness accelerates and eventually darkens the image to total black. Other good uses include an increasing soften (blur−out or −in) effect, or fade−in a background or foreground, instead of just jumping things into or out of the scene.

You can even imagine hooking up one blur to a background renderlayer, another inverted blur to a foreground renderlayer, and time feeding both. This node group would simulate someone focusing the camera lens.

Using Input Nodes

As your imagination runs wild, consider a few ideas that came to me just now on my couch: mixing a clouds texture with a time input to fog up a piece of glass or show spray paint building up on a wall. Consider mixing red and the soften with time (decreasing output) to show what someone sees when waking up from a hard hit on the head. Mix HSV input with a starfield image with time (decreasing output) to show what we might see someday as we accelerate our starship and experience red−shift.

As a user, you should know that we have arrived at the point where there are many ways to do the same thing in Blender. For example, an old way to make a slide show using Blender, you created multiple image textures, one image for each slide, and assigned them as texture channels to the material for the screen, then created a screen (plane) that filled the cameral view. Using a material ipo, you would adjust the Color influence of each channel at different frames, fading one in as the previous slide faded out. Whew! Rearranging slide and changing the timing was clunky but doable by moving the IPO keys. The *Node* way is to create an image input, one for each slide image. Using the Image input and Time nodes connected to an AlphaOver mixer is much simpler, clearer, and easier to maintain.

Previous: Manual/Compositing Nodes	Contents	*Next:* Manual/Compositing Nodes Output

User Manual: Contents | Guidelines | Blender Version **2.43**

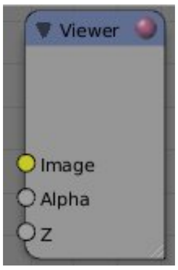
At any point, you may want to see or save the working image in progress, especially right after some operation by a node. Simply create another thread from the image output socket of the node to an Output node to see a mini−picture.

Only one Viewer and one Composite Node is active, which is indicated with a red sphere icon in the Node header. Clicking on Viewer Nodes makes them active. The active Composite Node is always the first, and you should only use one anyway.

Viewer

Panel: Node Editor Node Composition

Menu: Shift A â†’ Output â†’ Viewer



Viewer node

The Viewer node is a temporary, in−process viewer. Plug it in wherever you would like to see an image or value−map in your node−tree.

LMB click on the image to update it, if it wasn't done automatically. You can use as many of these as you would like.

Using the UV/Image Editor Window

The Viewer node allows results to be displayed in the UV/Image Editor. The image is facilitated by selecting the IM:Viewer Node on the window's header. The UV/Image Editor will display the image from the currently selected viewer node.

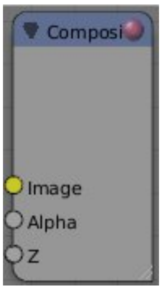
To save the image being viewed, use the Image−>Save As menu to save the image in a file.

The UV/Image Editor also has three additional options in its header to view Images with or without Alpha, or to view the Alpha or Z itself. Holding LMB in the Image display allows you to sample the values.

Composite

Panel: Node Editor Node Composition

Menu: Shift A â†’ Output â†’ Composite



Composite node

The Composite node is where the actual output from the compositor is connected to the renderer. Connecting a node to the Composite node will output the result of that node's full tree to the Renderer, leaving this node unconnected will result in a blank image. This node is updated after each render, but also if you change things in your node−tree (provided at least one finished input node is connected).

You can connect three channels: the actual RGBA image, the Alpha image, and the Z (depth) image. You should only have one Composite node in your map so that only one final image is rendered when the Do Composite button is pressed on the Render Options Animation panel (please read Node Composition if you haven't yet). Otherwise, unpredictable results may occur.

Saving your Composite Image

The RENDER button renders a single frame or image. Save your image using F3 or the File−>Save Image menu. The image will be saved using the image format settings on the Render panel.

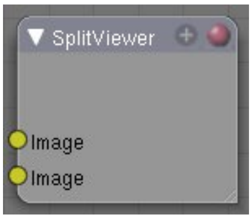
To save a sequence of images, for example if you input a movie clip or used a Time node, with each frame in its own file, use the ANIM button and its settings. If you might want to later overlay them, be sure to use an image format that supports an Alpha channel (such as PNG). If you might want to later arrange them front to back or create a depth of field effect, use a format that supports a Z−depth channel (such as EXR).

To save a composition as a movie clip (all frames in a single file), use an AVI or Quicktime format, and use the ANIM button and its settings.

SplitViewer Node

Panel: Node Editor Node Composition

Menu: Shift A â†' Output â†' SplitViewer



SplitViewer node

The SplitViewer node takes two images and displays one half of each on each side (top socket on the right half, bottom socket input on the left). Use this node for making side−by−side comparisons of two renderings/images, perhaps from different renderlayers or from different scenes. When transitioning between scenes, you want to be sure the stop action is seamless; use this node to compare the end of one scene with the beginning of another to ensure they align.

File Output Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Output â†’ File Output

File Output node

This node puts out an RGBA image, in the format selected, for each frame range specified, to the filename entered, as part of a frameset sequence. This means that the name of the file will be the name you enter plus a numeric frame number, plus the filename extension (based on format). Based on the format you choose, various quality/compression options may be shown.

To support subsequent arrangement and layering of images, the node can supply a Z−depth map. However, please note that only the OpenEXR image formats save the Z information.

The image is saved whenever Blender feels like it. Just kidding; whenever you press the Render button, the current frame image is saved. When you press the Anim button, the frameset sequence (specified in the Start and End frame) is saved.

This node saves you from doing (or forgetting to do) the Save Image after a render; the image is saved automagically for you. In addition, since this node can be hooked in anywhere in the noodle, you can save intermediate images automatically. Neat, huh?

Filespecs

As with all filename entries, use // at the beginning of the field to shorthand reference the current directory of the .blend file. You can also use the .. breadcrumb to go up a directory.

Previous: Manual/Compositing Nodes Input	Contents	*Next:* Manual/Compositing Nodes Color

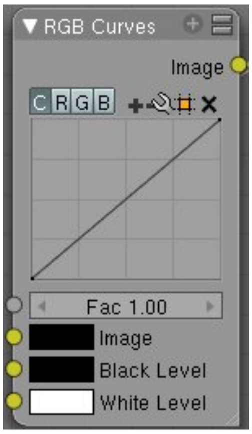
User Manual: Contents | Guidelines | Blender Version **2.43**

These nodes play with the colors in the image. They adjust the image's color intensity, adjust contrast and intensity, and, most importantly, mix two images together by color, transparency, or distance.

RGB Curves Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ RGB Curves

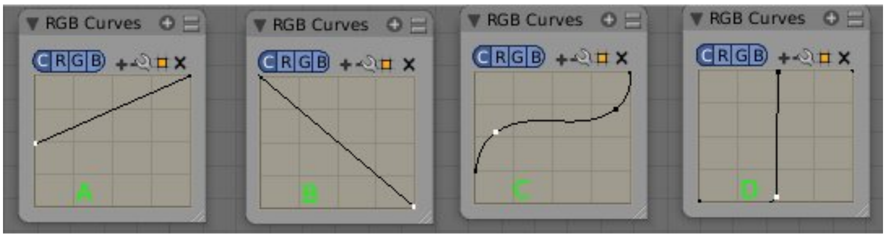


RGB Curves node

For each color component channel (RGB) or the composite (C), this node allows you to define a bezier curve that varies the input (across the bottom, or x−axis) to produce an output value (the y−axis). By default, it is a straight line with a constant slope, so that .5 along the x−axis results in a .5 y−axis output. Click and drag along the curve to create a control point and to change the curve's shape. Use the X to delete the selected (white) point.

Clicking on each C R G B component displays the curve for that channel. For example, making the composite curve flatter (by clicking and dragging the left−hand point of the curve up) means that a little amount of color will result in a lot more color (a higher Y value). Effectively, this bolsters the faint details while reducing overall contrast. You can also set a curve just for the red, and for example, set the curve so that a little red does not show at all, but a lot of red does.

Here are some common curves you can use to achieve desired effects:



A) Lighten **B**) Negative **C**) Decrease Contrast **D**) Posterize

Options

Fac
: How much the node should factor in its settings and affect the output

Black Level
: Defines the input color that is mapped to black. Default is black, which does not change the image.

White Level
: Defines the input color that is mapped to white. Default is white, which does not change the image.

The levels work exactly like the ones in the image viewer. Input colors are scaled linearly to match black/white levels.

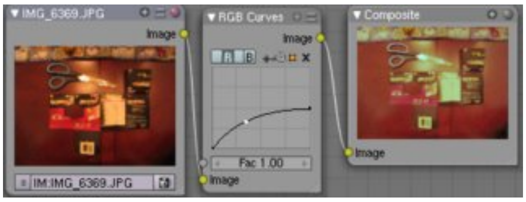
To define the levels, either use LMB on the color patch to bring up the color selection widget or connect some RGBA input to the sockets.

To only affect the value/contrast (not hue) of the output, set the levels to shades of gray. This is equivalent to setting a linear curve for C.

If you set any level to a color with a saturation greater than 0, the output colors will change accordingly, allowing for basic color correction or effects. This is equivalent to setting linear curves for R, G and B.

Examples

Color correction using Curves



Color correction with curves

In this example, the image has way too much red in it, so we run it through an RGB node and reduce the Red channel by about half.

We added a middle dot so we could make the line into a sideways exponential curve. This kind of curve evens out the amount of a color in an image as it reaches saturation. Also, read on for examples of the Darken and Contrast Enhancement curves.

Color correction using Black/White Levels

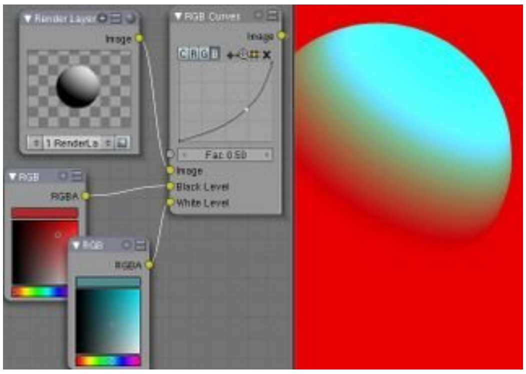
Color correction with Black/White Levels

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it's best to bring up an image viewer window showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zoomig in to pixel level if necessary. The result can be fine−tuned with the R,G, and B curves like in the previous example.

The curve for C is used to compensate the increased contrast that is a side−effect of setting Black and White Levels.

Effects



Changing colors

Curves and Black/White Levels can also be used to completely change the colors of an image.

Note that e.g. setting Black Level to red and White Level to blue does not simply substitute black with red and white with blue as the example image might suggest. Levels do color scaling, not substitution, but depending on the settings they can result in the described color substitution.

(What really happens when setting Black Level to pure red and White Level to pure blue is that the red channel gets inverted, green gets reduced to zero and blue remains unchanged.)

Because of this the results of setting arbitrary Black/White Levels or RGB curves is hard to predict, but can be fun to play with.

Mix Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ Mix

This node mixes a base image (threaded to the top socket) together with a second image (bottom socket) by working on the individual and corresponding pixels in the two images or surfaces. The way the output image is produced is selected in the drop−down menu. The size (output resolution) of the image produced by the mix node is the size of the base image. The alpha and Z channels are mixed as well.

Not one, not two, but count 'em, *sixteen* mixing choices include:

Mix
: The background pixel is covered by the foreground using alpha values.

Add
: The pixels are added together. *Fac* controls how much of the second socket to add in. Gives a bright result.
The "opposite" to Subtract mode.

Subtract
: Pixels are subtracted from one another. Gives a dark result.
The "opposite" to Add mode.

Multiply
: Returns a darker result than either pixel in most cases (except one of them equals white=1). Completely white layers do not change the background at all. Completely black layers give a black result.
The "opposite" to Screen mode.

Screen
: Both pixel values are inverted, multiplied by each other, the result is inverted again. This returns a brighter result than both input pixels in most cases (except one of them equals 0). Completely black layers do not change the background at all (and vice versa) − completely white layers give a white result.
The "opposite" of Multiply mode.

Overlay
: A combination of Screen and Multiply mode, depending on the base color.

Divide
: The background pixel (top socket) is divided by the second one: if this one is white (= 1.0), the first one isn't changed; the darker the second one, the brighter is the result (division by 0.5 − median gray − is same as multiplication by 2.0); if the second is black (= 0.0, zero−division is impossible!), Blender doesn't modify the background pixel.

Difference
: Both pixels are subtracted from one another, the absolute value is taken. So the result shows the distance between both parameters, black stands for equal colors, white for opposite colors (one is black, the other white). The result looks a bit strange in many cases. This mode can be used to invert parts of the base image, and to compare two images (results in black if they are equal).

Darken
: Both pixels are compared to each other, the smaller one is taken. Completely white layers do not change the background at all, and completely black layers give a black result.

Lighten
: Both parameters are compared to each other, the larger one is taken. Completely black layers do not change the image at all and white layers give a white result.

Dodge
: Some kind of inverted Multiply mode (the multiplication is replaced by a division of the "inverse"). Results in lighter areas of the image.

Burn

Some kind of inverted Screen mode (the multiplication is replaced by a division of the "inverse"). Results in darker images, since the image is *burned* onto the paper, er..image (showing my age).

Color

Adds a color to a pixel, tinting the overall whole with the color. Use this to increase the tint of an image.

Value

The RGB values of both pixels are converted to HSV values. The values of both pixels are blended, and the hue and saturation of the base image is combined with the blended value and converted back to RGB.

Saturation

The RGB values of both pixels are converted to HSV values. The saturation of both pixels are blended, and the hue and value of the base image is combined with the blended saturation and converted back to RGB.

Hue

The RGB values of both pixels are converted to HSV values. The hue of both pixels are blended, and the value and saturation of the base image is combined with the blended hue and converted back to RGB.

Color Channels

There are two ways to express the channels that are combined to result in a color: RGB or HSV. RGB stands for the Red,Green,Blue pixel format, and HSV stands for Hue,Saturation,Value pixel format.

A

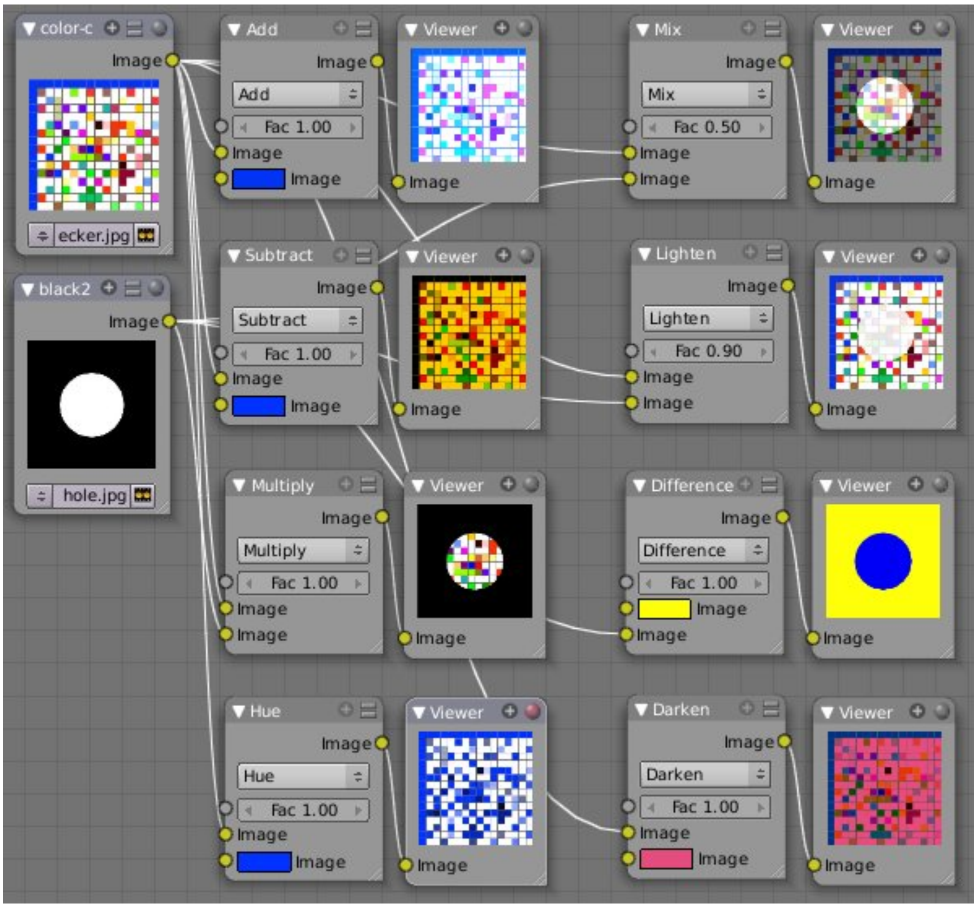
Click the green Alpha button to make the mix node use the Alpha (transparency) values of the second (bottom) node. If enabled, the resulting image will have an Alpha channel that reflects both images' channels. Otherwise, (when not enabled, light green) the output image will mix the colors by considering what effect the Alpha channel has of the base (top input socket) image. The Alpha channel of the output image is not affected.

Fac

The amount of mixing of the bottom socket is selected by the Factor input field (Fac:). A factor of zero does not use the bottom socket, where as a value of 1.0 makes full use. In Mix mode, 50:50 (0.50) is an even mix between the two, but in Add mode, .50 means that only half of the second socket's influence will be applied.

Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.

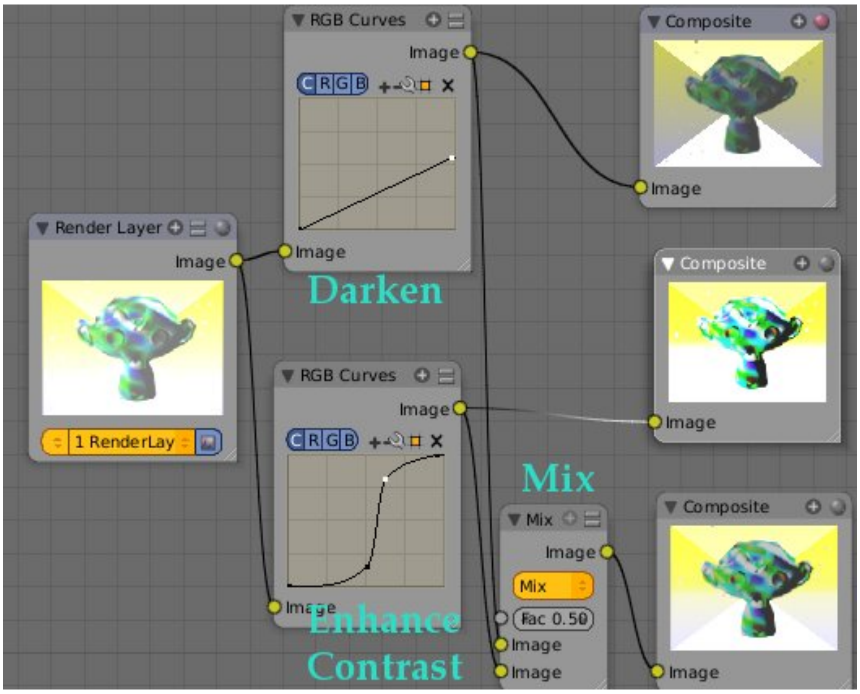


Some explanation of the mixing methods above might help you use the Mix node effectively:

- *Add* − adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.
- *Subtract* − Taking Blue away from white leaves Red and Green, which combined make Yellow (and you never thought you'd need a color wheel again, eh?). Taking Blue away from Purple leaves Red. Use this to de−saturate an image. Taking away yellow makes an image bluer and more depressing (fun to do, if you're emo).
- *Multiply*: Black (0.00) times anything leaves black. Anything times White (1.00) is itself. Use this to mask out garbage, or to colorize a black−and−white image.
- *Hue*: Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').
- *Mix*: Combines the two images, averaging the two
- *Lighten*: Like bleach, makes your whites whiter. Use with a mask to lighten up a little.
- *Difference*: Kinda cute in that it takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two verrry similar images to see what had been done to one to make it the other; sorta like a change log for images. You can use this to see a watermark you have placed in an image for theft detection.
- *Darken*, with the colors set here, is like looking at the world through rose−colored glasses (sorry, I just couldn't resist).

Contrast Enhancement using Mix

Here is a small map showing the effects of two other common uses for the RGB Curve: **Darken** and **Contrast Enhancement**. You can see the effect each curve has independently, and the combined effect when they are **mixed** equally.



Example node setup showing "Darken", "Enhance Contrast" and "Mix" nodes for composition.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast. Other paint programs usually provide a slider type of control, but Blender, ah the fantastic Blender, provides a user−definable curve to provide precise control.

In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB 'S' curve scales the output such that middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better. And NOBODY wants a cranky monkey on their hands.

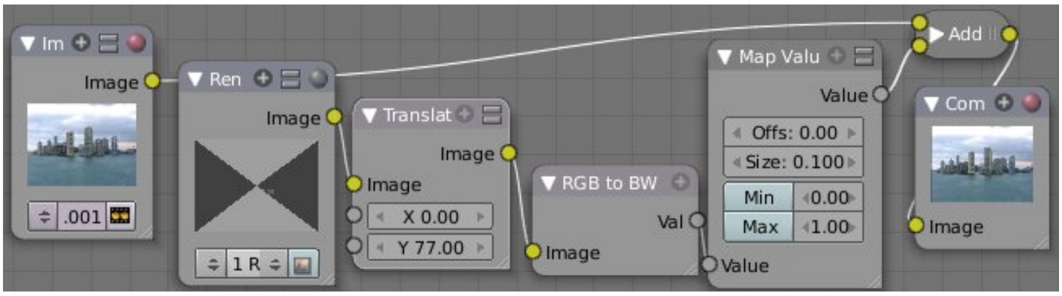
Using Mix to Watermark images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perciptable except in just the right light. Probably the first form of subliminal advertising. Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting service, or simply to track their image's proliferation throughout the web. Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding Your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it's up to you. In the example below, we are encoding the watermark in a specific location in the image using the Translate node; this helps later because we only have to look in a specific location for the mark. We then use the RGB to BW node to convert the image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one−tenth of its original intensity. The Add node adds the corresponding pixels, make the ones

containing the mark ever−so−slightly brighter.



Embedding your mark in an Image using a Mark and Specific Position

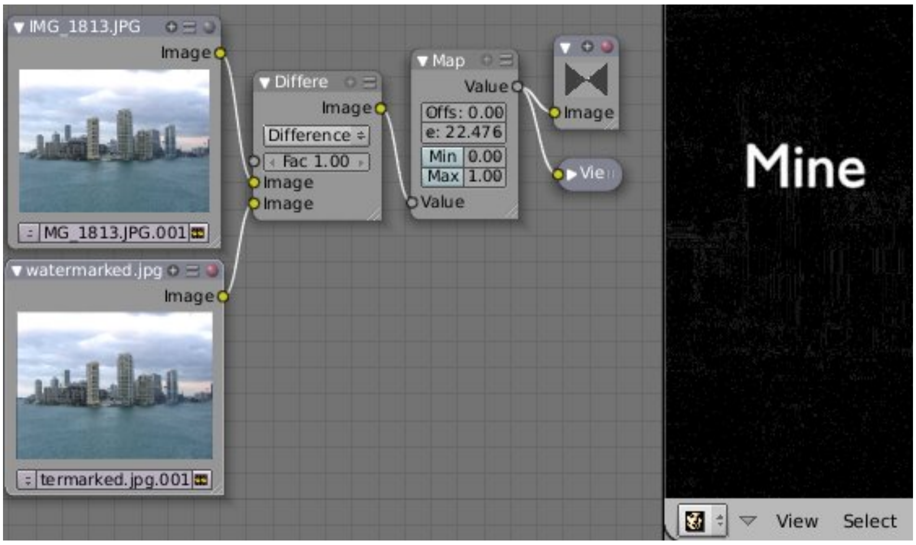
Of course, if you *want* people to notice your mark, don't scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Additional uses

You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface, or as a special effect to make words appear on the posessed girl's forearm. yuk.

Decoding an Image for your Watermark

When you see an image that you think might be yours, use the node map below to compare it to your stock image (pre−watermarked original). In this map, the Mix node is set to Difference, and the Map Value node amplifies any difference. The result is routed to a viewer, and you can see how the original mark stands out, clear as a bell:



Checking an image for your watermark

Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding map in one scene, and the decoding map in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to JPEG compression of 50%.

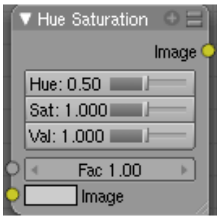
Using Dodge and Burn (History Lesson)

Use the dodge and burn mix methods in combination with a mask to affect only certain areas of the image. In the old darkroom days, when yes, I actually spent hours in a small stinky room bathed in soft red light, I used a circle cut out taped to a straw to dodge areas of the photo as the exposure was made, casting a shadow on the plate and thus limiting the light to a certain area. To do the opposite, I would burn in an image by holding a mask over the image. The mask had a hole in it, letting light through and thus 'burning' in the image onto the paper. The same equivalent can be used here by mixing an alpha mask image with your image using a dodge mixer to lighten an area of your photo. Remember that black is zero (no) effect, and white is one (full) effect. And by the way, ya grew to like the smell of the fixer, and with a little soft music in the background and the sound of the running water, it was very relaxing. I kinda miss those dayz.

Hue Saturation Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ Hue Saturation



As an alternative to RGB editing, color can be thought of as a mix of Hues, namely a normalized value along the visible spectrum from infra−red to ultra violet (the rainbow, remember "Roy G. Biv"). The amount of the color added depends on the saturation of that color; the higher the saturation, the more of that pigment is added. Use the saturation slider of this node to "bring out" the colors of a washed out image.

This node takes an input image and runs the color of the image (and the light it reflects and radiates) 'up' through a factor (0.0−1.0) and applies a saturation of color effect of a hue to the image:

Hue:
: The **Hue** slider specifies how much to shift the hue of the image. Hue 0.5 (in the middle) does not shift the hue or affect the color of the image. As Hue shifts left, the colors shift as more cyan is added; a blue image goes bluer, then greener, then yellow. A red image goes violet, then purple, blue, and finally teal. Shifting right (increasing Hue from 0.5 to 1.0) introduces reds and greens. A blue image goes purple, plum, red, orange, and then yellow. A red image goes golden, olive, green, and cyan.

Sat:
: **Saturation** affect the amount of pigment in the image. A saturation of 0 actually *removes* hues from the color, resulting in a black−and−white grey scale image. A saturation of 1.0 blends in the hue, and 2.0 doubles the amount of pigment and brings out the colors.

Val:
: **Value** affects the overall amount of the color in the image. Increasing value makes an image lighter; decreaing value shifts an image darker.

Fac:
: **Factor** determines how much this node affects the image. A factor of 0 means that the input image is not affected by the Hue and Saturation settings. A factor of 1 means they rule, with .5 being a mix.

Hue Saturation tips

Some things to keep in mind, that might help you use this node better:

Hues are vice−versa.
A Blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (Recall that white, minus blue, equals yellow). A Yellow image, with a Hue setting at 0 or 1, is blue.

Hue and Saturation work together.
So, a Hue of .5 keeps the blues the same shade of blue, but the saturation slider can deepen or lighten the intensity of that color.

Gray & White are a neutral hue.
A grey image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with the Val slider. This applies for all shades of grey, from black to white; wherever the values are equal.

Changing the effect over time.
The Hue and Saturation values are set in the node by the slider, but you can feed a Time input into the Factor to bring up (or down) the effect change over time.

Tinge

This HSV node simply shifts hues that are already there. To colorize a gray image, or to ADD color to an image, use a mix node to add in a static color from an RGB input node with your image.

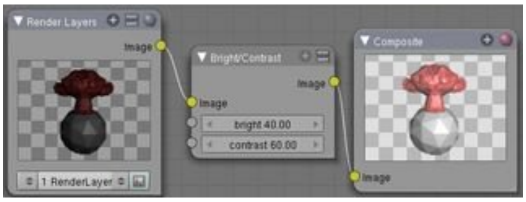
HSV Example

Here, the image taken by a cheap digital camera in poor lighting at night using a flash (can we do it any worse, eh?) is adjusted by decreasing the Hue (decreasing reds and revealing more blues and greens), decreasing Saturation (common in digital cameras, and evens out contrast) and increasing Value (making it all lighter).

Bright/Contrast

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ Bright/Contrast



A basic example

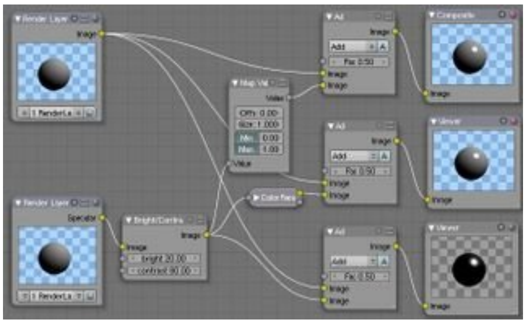
Bright

A multiplier−type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast

A scaling type factor by which to make brighter pixels brighter but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

Notes



It is possible that this node will put out a value set that has values beyond normal range, i.e. values > 1 or < 0. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a ColorRamp node (with all normal defaults). Either of these nodes will scale the values back to normal range. In the example image, we want to amp up the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has valued much less than 1 in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the ColorRamp produces the desired effect.

Gamma

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ Invert

The reason for applying gamma correction to the final render is to correct lighting issues. Lighting issues that are corrected by the gamma correction are light attenuation with distance, light falloff at terminators and light and shadows superpositions. Simply think about the renderer as a virtual camera. By applying a gamma correction to your render, you are just replicating what digital camera do with the photos. Digital camera gamma correct their photos so you do the same thing. The gamma correction is, indeed, 0.45. Not 2.2.

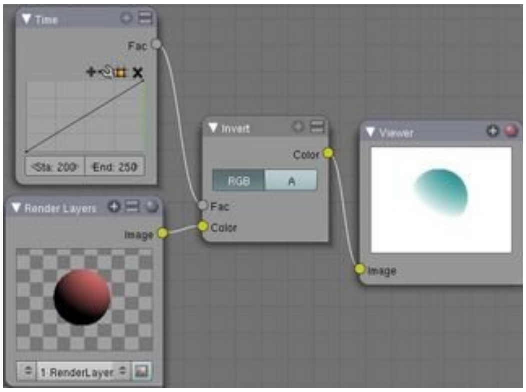
But the reverse gamma correction on the textures and colors have another very important consequence when you are using rendering techniques such as radiosity or GI. When doing the GI calculations, all textures and colors are taken to mean reflectance. If you do not reverse gamma correct your textures and colors, then the GI render will look way too bright because the reflected colors are all way too high and thus a lot more light is bouncing around than it should.

Gamma correction in Blender enters in a few places. The first is in this section with the nodes, both this node and the Tonemap node, and the second is in calculating Radiosity. In the noodle to the left, the split viewer shows the before and after effect of applying a gamma correction.

Invert

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ Invert



This handy node inverts the colors in the input image, producing a negative.

Options

Factor
: Controls the amount of influence the node exerts on the output image

Color
: The input image. In this case, a red sphere on a black transparent background

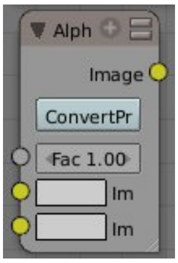
RGB
: Invert the colors from white. In this example, red inverted is cyan (teal).

A
: Invert the alpha (transparency) channel as well. Handy for masking.

AlphaOver Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ AlphaOver

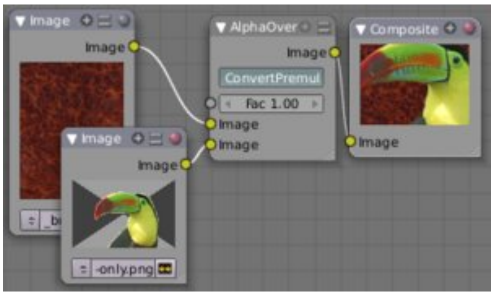


AlphaOver node

Use this node to layer images on top of one another. This node takes two images as input, combines them by a factor, and outputs the image. Connect the Background image to the top input, and the foreground image to the lower input. Where the foreground image pixels have an alpha greater than 0 (namely, have some visibility), the background image will be overlaid.

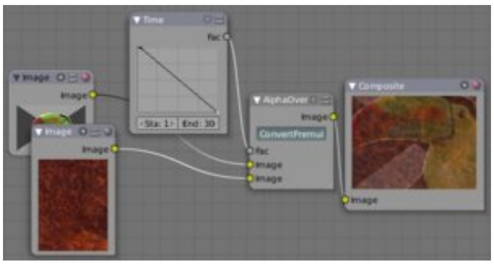
Use the Factor slider to 'merge' the two pictures. A factor less than 1.00 will make the foreground more transparent, allowing the background to bleed through.

Examples



Assembling a composite Image using AlphaOver

In this example, an image of a Toucan is superimposed over a wooden background. Use the PreMultiply button when the foreground image and background images have a combined Alpha that is greater than 1.00; otherwise you will see an unwanted halo effect. The resulting image is a composite of the two source images.



Animated See−Through/Sheer SFX using AlphaOver − Frame 11

In this example, we use the Factor control to make a sheer cloth or onion−skin effect. You can animate this effect, allowing the observer to 'see−through' walls (or any foreground object) by hooking up a Time node to feed the Factor socket as shown below. In this example, over the course of 30 frames, the Time node makes the AlphaOver node produce a picture that starts with the background wood image, and slowly bleeds through the Toucan. This example shows frame 11 just as the Toucan starts to be revealed.

AlphaOver does not work on the colors of an image, and will not output any image when one of the sockets is unconnnected.

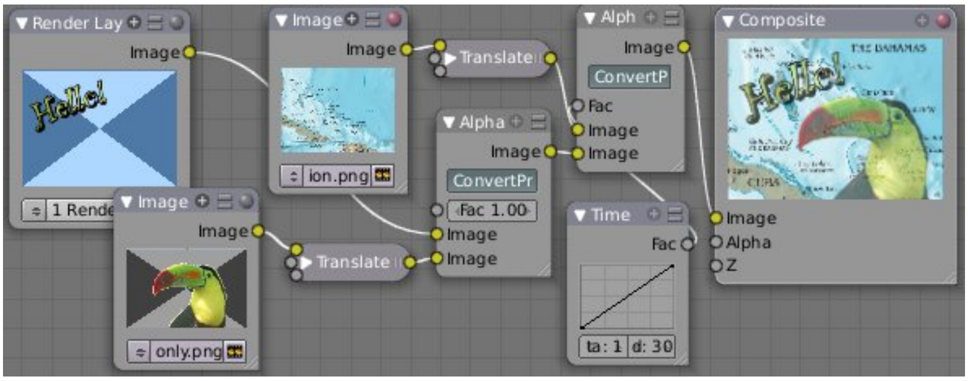
Strange Halos or Outlines

To clarify the premultiplied−alpha button: An alpha channel has a value of between 0 and 1. When you make an image transparent (to composite it over another one), you are really multiplying the RGB pixel values by the alpha values (making the image transparent (0) where the alpha is black (0), and opaque (1) where it is white (1)).

So, to composite image A over image B, you get the alpha of image A and multiply it by image A, thus making the image part of A opaque and the rest transparent. You then inverse the alphas of A and multiply image B by it, thus making image B transparent where A is opaque and vice versa. You then add the resultant images and get the final composite.

A pre−multiplied alpha is when the image (RGB) pixels are already multiplied by the alpha channel, therefore the above compositing op doesn't work too well, and you have to hit 'convert pre−mult'. This is only an issue in semi transparent area, and edges usually. The issue normally occurs in Nodes when you have combined, with alpha, two images, and then wish to combine that image with yet another image. The previously combined image was previously multiplied (pre−mult) and needs to be converted as such (hence, *Convert PreMul*).

If you don't pay attention and multiply twice, you will get a white or clear halo around your image where they meet, since your alpha value is being squared or cubed. It also depends on whether or not you have rendered your image as a pre−mult, or straight RGBA image.

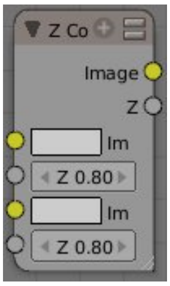


Layering Images using AlphaOver Premul

Z−Combine Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Color â†’ Z Combine

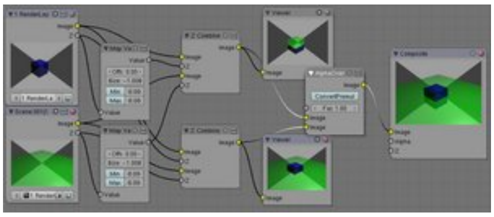


Z Combine node

The Z−Combine node takes two images and two Z−value sets as input. It overlays the images using the provided Z values to detect which parts of one image are in front of the other. If both Z values are equal, it uses the top image. It puts out the combined image, with the combined Z−depth map, allowing you to thread multiple Z−combines together.

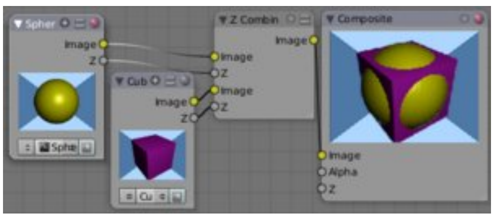
Z−Combine chooses whichever Z−value is less when deciding which image pixel to use. Normally, objects are in front of the camera and have a positive Z value. If one Z−value is negative, and the other positive, Z−Combine will use the image corresponding to the negative value. You can think of a negative Z value as being behind the camera. When choosing between two negative Z−values, Z−Combine will use whichever is more negative.

Alpha values carry over from the input images. Not only is the image pixel chosen, but also its alpha channel value. So, if a pixel is partially or totally transparent, the result of the Z−Combine will also be partially transparent; in which case the background image will show through the foreground (chosen) pixel. Where there are sharp edges or contrast, the alpha map will automatically be anti−aliased to smooth out any artifacts. Hoever, you can obtain this by making an AlphaOver of two Z−Combine, one normal, the other having inverted (reversed?) Z−values as inputs, obtained using for each of them a MapValue node with a Size field set to −1.0:



Alpha and Z−Combine node.

Examples



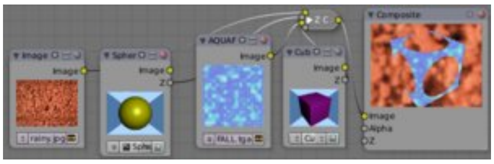
Choosing closest pixels

In the example to the right, render output from two scenes are mixed using the Z−Offset node, one from a sphere of size 1.30, and the other a cube of size 1.00. The sphere and square are located at the same place. The cube is tipped forward, so the corner in the center is closer to the camera than the sphere surface; so Z−Offset chooses to use the cube's pixels. But the sphere is slightly larger (a size of 1.30 versus 1.00), so it does not fit totally 'inside' the cube. At some point, as the cube's sides recede back away from the camera, the sphere's sides are closer. When this happens, Z−offset uses the sphere's pixels to form the resulting picture.

This node can be used to combine a foreground with a background matte painting. Walt Disney pioneered the use of multi−plane mattes, where three or four partial mattes were painted on glass and placed on the left and right at different Z positions; mininal camera moves to the right created the illusion of depth as Bambi moved through the forest.

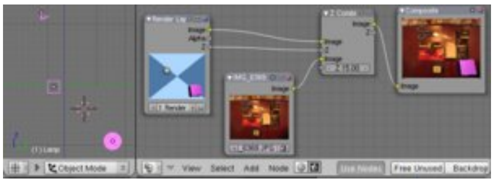
Valid Input

Z Input Sockets do not accept fixed values; they must get a vector set (see Map Value node). Image Input Sockets will not accept a color, since it does not have UV coordinates.



Mix and Match Images

You can use Z−Combine to merge two images as well, using the Z−values put out by two renderlayers. Using the Z−values from the sphere and cube scenes above, but threading different images, yields the example to the right.



Z−Combine in action

In this noodle (you may click the little expand−o−matic icon in the bottom right to view it to full size), we mix a render scene with a flat image. In the side view of the scene, the purple cube is 10 units away from camera, and the gray ball is 20. The 3D cursor is about 15 units away from camera. We Z−in the image at a location of 15, thus inserting it in−between the cube and the ball. The resulting image appears to have the cube on the table.

Invisible Man Effect

If you choose a foreground image which has a higher Alpha than the background, and then mix the Z−combine with a slightly magnified background, the outline of the transparent area will distort the background, enough to make it look like you are seeing part of the background through an invisible yet Fresnel−lens object.

Contents Next: Manual/Compositing Nodes Vector

Previous: Manual/Compositing Nodes
Output

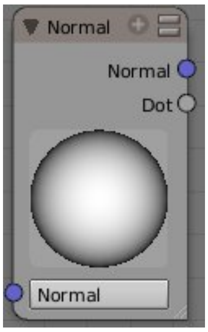
User Manual: Contents | Guidelines | Blender Version **2.43**

Vector nodes manipulate information about how light interacts with the scene, multiplying vector sets, and other wonderful things that normal humans barely comprehend (except math geniuses, who may not be considered 'normal'). Even if you aren't a math wiz, you'll find these nodes to be very useful.

Normal Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Vector â†’ Normal



Normal node

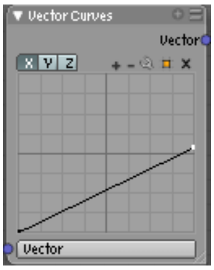
The Normal node generates a normal vector and a dot product. Click and Drag on the sphere to set the direction of the normal.

This node can be used to input a new normal vector into the mix. For example, use this node as an input to a Color Mix node. Use an Image input as the other input to the Mixer. The resulting colorized output can be easily varied by moving the light source (click and dragging the sphere).

Vector Curves Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Vector Ops â†’ Vector Curves



The Vector Curves node maps an input vector image's x, y, and z components to a diagonal curve. The three channels are accessed via the X, Y, and Z buttons at the top of the node. Add points to the curve by clicking on it.

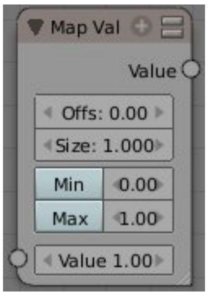
Note that dragging a point across another will switch the order of the two points (e.g. if point A is dragged acros point B, then point B will become point A and point A will become point B).

Use this curve to slow things down or speed them up from the original scene.

Map Value Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Vector â†’ Map Value



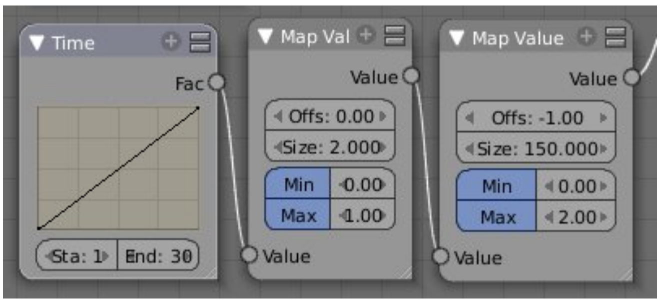
Map Value node

Map Value node is used to scale, offset and clamp values (value refers to each vector in the set). The formula for how this node works is:

- Offs will add a number to the input value
- Size will scale (multiply) that value by a number
- By clicking Min/Max you can set the minimum and maximum numbers to clamp (cut off) the value too. Min and Max must be individually enabled by LMB clicking on the label for them to clamp. Shift LMB on the value to change it.
    - ◊ If Min is enabled and the value is less than Min, set the ouput value to Min
    - ◊ If Max is enabled and the input value is greater than Max, set the ouput value to Max

This is particularly useful in achieving a depth−of−field effect, where you can use the Map Value node to map a Z value (which can be 20 or 30 or even 500 depending on the scene) to to range between 0−1, suitable for connecting to a Blur node.

Using Map Value to Multiply values



You can also use the map value node to multiply values to achieve an output number that you desire. In the mini−map to the right, the Time node ouputs a value between 0.00 and 1.00 evenly scaled over 30 frames. The *first* Map Value node multiplies the input by 2, resulting in an output value that scales from 0.00 to 2.00 over 30 frames. The *second* Map Value node subtracts 1 from the input, giving working values between −1.00 and 1.00, and multiplies that by 150, resulting in an output value between −150 and 150 over a 30−frame

sequence.

Previous: Manual/Compositing Nodes Color	Contents	*Next:* Manual/Compositing Nodes Filters

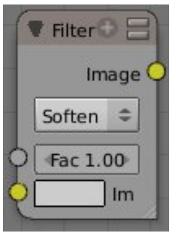
User Manual: Contents | Guidelines | Blender Version **2.44**

Filters process the pixels of an image to highlight additional details or perform some sort of post−processing effect on the image.

Filter Node

Panel: Node Editor Node Composition

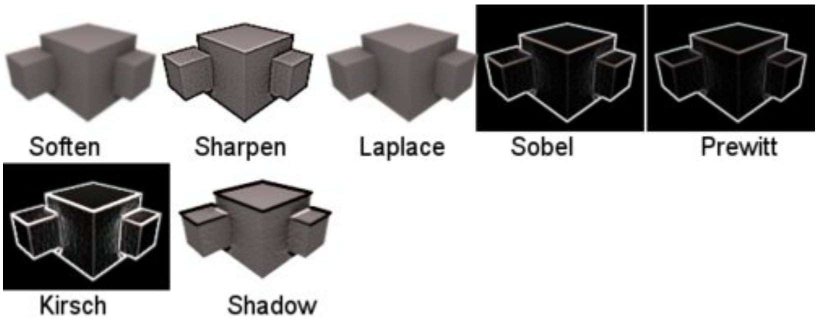
Menu: Shift A â†’ Filters â†’ Filter



Filter node

The Filter node implements various common image enhancement filters. The supported filters are, if not obvious, named after the mathematical genius who came up with them:

- Soften − Slightly blurs the image.
- Sharpen − Increases the contrast, especially at edges
- Laplace − Softens around edges
- Sobel − Creates a negative image that highlights edges
- Prewitt − Tries to do Sobel one better.
- Kirsch − Improves on the work done by those other two flunkies, giving a better blending as you approach an edge.
- Shadow − Performs a relief emboss/bumpmap effect, darkening outside edges.



The Filter node has seven modes, shown here.

The Soften, Laplace, Sobel, Prewitt and Kirsch all perform edge−detection (in slightly different ways) based on vector calculus and set theory equations that would fill six blackboards with gobbledy gook. Recommended reading for insomniacs.

Blur Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Filters â†’ Blur

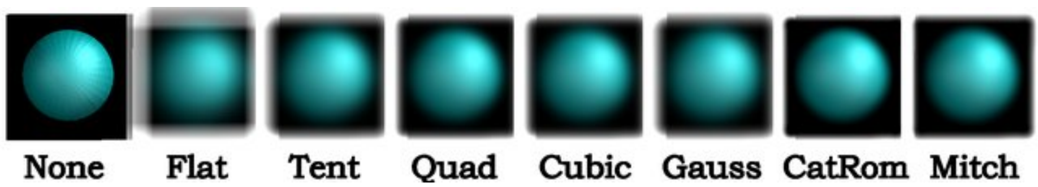
Blur node

The Blur node blurs an image, using one of seven blur modes (set using the upper−left popup button), and a radius defined by the X and Y number buttons. By default these are set to zero, so to enable the node you must set one or both to a value greater then 0. You can optionally connect a value image to the Size input node, to control the blur radius with a mask. The values must be mapped between 0−1 for best effect, as they will be multiplied with the X and Y number button values.

Options

The X and Y values are the number of pixels over which to spread the blur effect.

The Bokeh button (only visible as Bok or Bo on some screen setups) will force the blur node to use a circular blur filter. This gives higher quality results, but is slower then using a normal filter. The Gam button (for "gamma") makes the Blur node gamma−correct the image before blurring it.



Blur node blur modes using 15% of image size as XY, no Bokeh/Gamma. Click expand to see details

The difference between them is how they handle sharp edges and smooth gradients and preserve the highs and the lows. In particular (and you may have to closely examine the full−resolution picture to see this):

Flat just blurs everything uniformly
Tent preserves the high and the lows better making a linear falloff
Quadratic and CatRom keep sharp−contrast edges crisp
Cubic and Mitch preserve the highs but give almost a out−of−focus blur while smoothing sharp edges

Example

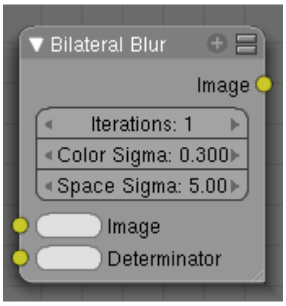
An example blend file, in fact the one used to create the image above, is available here. The .blend file takes one image from the RenderLayer "Blurs" and blurs it while offsetting it (Translate) and then combining it

(AlphaOver) to build up the progressive sequence of blurs. Play with the Value and Multiply nodes to change the amount of blurring that each algorithm does.

Bilateral Blur Node

Panel: Node Editor Node Composition

Menu: Shift A â†' Filters â†' Bilateral Blur



Blur node

The bilateral blur node performs a high quality adaptive blur on the source image. It can be used for various purposes like:

smoothing results from blenders raytraced ambient occlusion
smoothing results from various unbiased renderers,
to fake some performance−heavy processes, like blurry refractions/reflections, soft shadows,
to make non−photorealistic compositing effects.

Inputs

Bilateral blur has 2 inputs:

Image, for the image to be blurred.
Determinator, which is non−obligatory, and is used only if connected.

if only 1st input is connected, the node blurs the image depending on the edges present in the source image. If the Determinator is connected, it serves as the source for defining edges/borders for the blur in the image. This has great advantage in case the source image is too noisy, but normals in combination with zbuffer can still define exact borders/edges of objects.

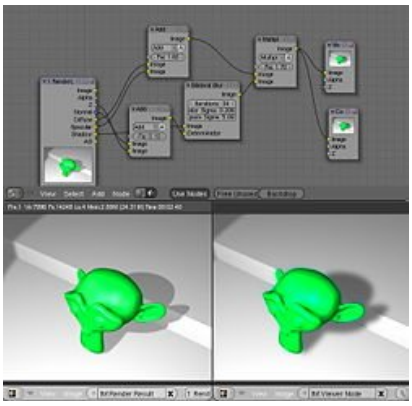
Options

Iterations − defines how many times should the filter perform the operation on the image. It practically defines the radius of blur.

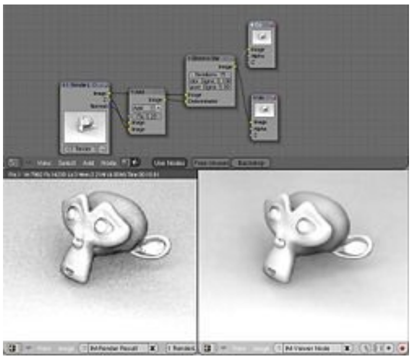
Color Sigma − defines the threshold for which color differences in the image should be taken es edges.

Space sigma − is a fine−tuning variable for blur radius.

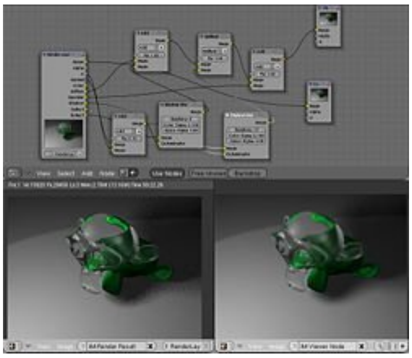
Examples



Bilateral smoothed buffered shadow



Bilateral smoothed AO

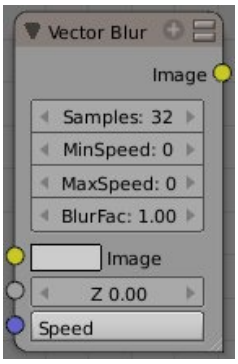


Bilateral faked blurry refraction+smoothed reytraced soft shadow

Vector (Motion) Blur Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Filters â†’ Vector Blur



Vector Blur node

Motion blur is the effect of objects moving so fast they blur. Because CG animations work by rendering individual frames, they have no real knowledge of what was where in the last frame, and where it is now.

In Blender, there are two ways to produce motion blur. The first method (which produces the most correct results) works by rendering a single frame up to 16 times with slight time offsets, then accumlating these images together; this is called Motion Blur and is activated on the Render panel. The second (and much faster) method is the Compositor node Vector Blur.

To use, connect the appropriate passes from a Render Result node.

Note

Make sure to enable the Speed (called Vec) pass in the Render Layers panel for the render layer you wish to perform motion blur on.

Maximum Speed: Because of the way vector blur works, it can produce streaks, lines and other artifacts. These mostly come from pixels moving too fast; to combat these problems, the filter has minimum and maximum speed settings, which can be used to limit which pixels get blurred (e.g. if a pixel is moving really, really fast but you have maximum speed set to a moderate amount, it won't get blurred).

Minimum Speed: Especially when the camera itself moves, the mask created by the vectorblur node can become the entire image. A very simple solution is to introduce a small threshold for moving pixels, which can efficiently separate the hardly−moving pixels from the moving ones, and thus create nice looking masks. You can find this new option as 'min speed'. This minimum speed is in pixel units. A value of just 3 will already clearly separate the background from foreground.

Hint

You can make vector blur results a little smoother by passing the Speed pass through a blur node (but note that this can make strange results, so it's only really appropriate for still images with lots of motion blur).

Examples

An in−depth look at how to use the Vector Blur node can be found here.

As far as we know, this node represents a new approach to calculating motion blur. Use vector blur in compositing with confidence instead of motion blur. In face, when compositing images, it is necessary to use vector blur since there isn't "real" motion. In this example blend file, you will find a rigged hand reaching down to pick up a ball. Based on how the hand is moving (those vectors), the image is blurred in that direction. The fingers closest to the camera (the least Z value) are blurred more, and those farther away (the

forearm) is blurred the least.

Known Bugs

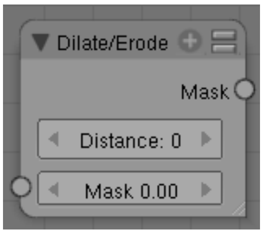
V.: 2.44

Does not work when reading from a multilayer OpenEXR sequence set

Dilate/Erode Node

Panel: Node Editor Node Composition

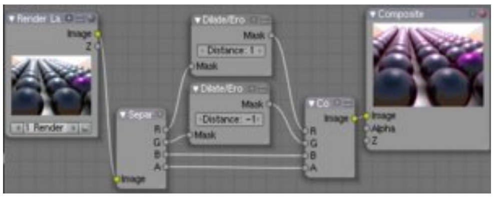
Menu: Shift A â†’ Filters â†’ Dilate/Erode



Dilate/Erode node

This node blurs individual color channels. The color channel (or a black and white image) is connected to the Mask input socket, and the Distance is set manually (by clicking on the arrows or the value) or automatically from a value node or a time−and−map−value noodle. A positive value of Distance expands the influence of a pixel on its surrounding pixels, thus blurring that color outward. A negative value erodes its influence, thus increases the constrast of that pixel relative to its surrounding pixels, thus sharpening it relative to surrounding pixels of the same color.

Example



Magenta tinge

In the above example image, we wanted to take the rather boring array of ball bearings and spruce it up; make it hot, baby. So, we dilated the red and eroded the green, leaving the blue alone. If we had dilated both red and green...(hint: red and green make yellow). The amount of influence is increased by increasing the Distance values. [Blend file available here.]

Defocus

Panel: Node Editor Node Composition

Menu: Shift A â†’ Filters â†’ Defocus

This single node can be used to emulate depth of field using a postprocessing method. It can also be used to blur the image in other ways, not necessarily based on 'depth' by connecting something other than a Zbuffer. In essence, this node blurs areas of an image based on the input zbuffer map/mask.

Camera Settings

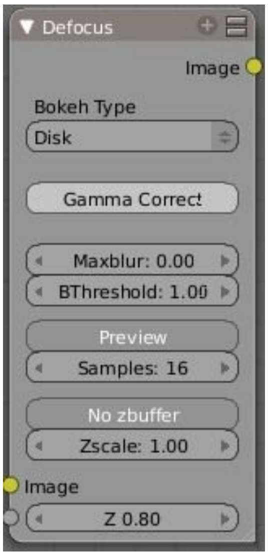
DofDist setting for the camera.

The Defocus node uses the actual camera data in your scene if supplied by a RenderLayer node.

To set the point of focus, the camera now has a DoFDist parameter, which is shorthand for Depth of Field Distance. Use this camera parameter to set the focal plane of the camera (objects DoFDist away from the camera are in focus). Set DofDist in the main Camera edit panel; the button is right below the Lens value button.

To make the focal point visible, enable the camera Limits option, the focal point is then visible as a yellow cross along the view direction of the camera.

Node Inputs



Defocus node

The node requires two inputs, an image and a zbuffer, the latter does not need to be an actual zbuffer, but can also be another (grayscale) image used as mask, or a single value input, for instance from a time node, to vary the effect over time.

Node Setting

The settings for this node are:

Bokeh Type menu
: Here you set the number of iris blades of the virtual camera's diaphragm. It can be set to emulate a perfect circle (Disk) or it can be set to have 3 (Triangle), 4 (Square), 5 (Pentagon), 6 (Hexagon), 7 (Heptagon) or 8 blades (Octagon). The reason it does not go any higher than 8 is that from that point on the result tends to be indistinguishable from a Disk shape anyway.

Rotate
: This button is not visible if the Bokeh Type is set to Disk. It can be used to add an additional rotation offset to the Bokeh shape. The value is the angle in degrees.

Gamma Correct
: Exactly the same as the Gamma option in Blender's general Blur node (see #Blur Node). It can be useful to further brighten out of focus parts in the image, accentuating the Bokeh effect.

Defocus node using Z−Buffer

fStop
: This is the most important parameter to control the amount of focal blur: it simulates the aperture *f* of a real lens(' iris) − without modifying the luminosity of the picture, however! Let's remind that the more open is *f*, the more closed is the iris and the wider is the focus field (the area in witch the sharpness of the picture is considered to be good enough). The default value 128 is assumed to be infinity, everything in perfect focus. Half the value will double the amount of blur. This button is not available if No zbuffer is enabled.

Maxblur
: Use this to limit the amount of blur of the most out of focus parts of the image. The value is the maximum blur radius allowed. This can be useful since the actual blur process can sometimes be very slow, the more blur, the slower it gets. So setting this value can help bring down processing times, like for instance when the world background is visible, which in general tends to be the point of maximum blur (not always true, objects very close to the lens might be blurred even more). The default value of 0 means there is no limit to the maximum blur amount.

BThreshold
: The defocus node is not perfect, and some artefacts may occur. One such example is on in−focus objects on a blurred background, which has a tendency to bleed into the edges of the sharp object. Worst case scenario is an object in focus against the very distant world background, the differences in distance are very large and the result can look quite bad. The node tries to prevent this from occurring by testing that the blur difference between pixels is not too large, the value set here controls how large that blur difference may be to consider it 'safe'. This is all probably quite confusing, and fortunately, in general, there is no need to change the default setting of 1. Only try changing it if you experience problems around any in−focus object.

Preview
: As already mentioned, processing can take a long time. So to help make editing parameters somewhat 'interactive', there is a preview mode which you can enable with this button. Preview mode will render the result using a limited amount of (quasi)random samples, which is a *lot* faster than the 'perfect' mode used otherwise. The sampling mode also tends to look like something a typical raytracer like yafray might produce, grainy, noisy pictures (though the more samples you use, the less noisy the result). This option is on by default, play around with the other parameters until you are happy with

the results, only then disable the preview mode for the final render.

Samples
: Only visible when Preview is set. Sets the amount of samples to use to sample the image. The higher, the smoother the image, but also the longer the processing time. For preview, the default of 16 samples should be sufficient and is also the fastest.

No zbuffer
: Sometimes you might want to have more control to blur the image, for instance only blur one object while leaving everything else alone, or the other way around, or you want to blur the whole image uniformly all at once.
The node therefore allows you to use something other than an actual zbuffer as the Z input. For instance, you could connect an image node and use a grayscale image where the color designates how much to blur the image at that point, where white is maximum blur and black is no blur. Or you could use a time node to uniformly blur the image, where the time value controls the maximum blur for that frame. It may also be used for a possibly slightly better DoF blur as well, by using a fake depth shaded image instead of a zbuffer. (A typical method to create the fake depth shaded image is by using a linear blend texture for all objects in the scene or by using the 'fog/mist' fake depth shading method). This also has the advantage that the fake depth image can have anti−aliasing which is not possible with a real zbuffer.
No zbuffer will be enabled automatically whenever you connect a node that is not image based (e.g. time node/value node/etc)

Zscale
: Only visible when No zbuffer enabled. When No zbuffer is used, the input is used directly to control the blur radius. And since usually, a texture is only in the range 0 to 1, it's range is too narrow to control the blur properly. This parameter can be used to expand the range of the input, or for that matter, narrow it as well, by setting it to a value less than one. So for No zbuffer, this parameter therefore then becomes the main blur control, similar to fStop when you *do* use a zbuffer.

Examples

In this blend file example, the ball array image is blurred as if it was taken by a camera with a f−stop of 2.8 resulting in a farily narrow depth of field centered on 7.5 blender units from the camera. As the balls receed into the distance, they get blurrier.

Hints

Preview
: In general, use preview mode, change parameters to your liking, only then disable preview mode for the final render. This node is compute intensive, so watch your console window, and it will give you status as it computes each render scan line.

Edge Artifacts

For minimum artifacts, try to setup your scene such that differences in distances between two objects that may visibly overlap at some point are not too large. Keep in mind that this is not 'real' DoF, only a postprocessing simulation. Some things cannot be done which would be no problem for real DoF at all. A typical example is a scene with some object very close to the camera, and the camera focusing on some point far behind it. In the real world, using shallow depth of field, it is not impossible for nearby objects to become completely invisible, in effect allowing the camera to see 'behind' it. This is simply not possible to do with the current postprocessing method in a single pass. If you really want or need to emulate such an effect, the only way to do this is to split up your scene into nearby and far object(s), rendering in two passes, then postprocess each separately and combine the results.

Aliasing at Low f−Stop Values

At very low values, less than 5, the node will start to remove any oversampling and bring the objects at DoFDist very sharply into focus. If the object is against a constrasting background, this may lead to visible stairstepping (aliasing) which OSA is designed to avoid. If you run into this problem:

- ◊ Do your own OSA by rendering at twice the intended size and then scaling down, so that adjacent pixels are blurred togther
- ◊ Use the blur node with a setting of 2 for x and y
- ◊ Set DoFDist off by a little, so that the object in focus is blurred by the tiniest bit.
- ◊ Use a higher f−Stop, which will start the blur, and then use the Z socket to a Map Value to a Blur node to enhance the blur effect.
- ◊ Rearrange the objects in your scene to use a lower−contrast background

No ZBuffer

A final word of warning, since there is no way to detect if an actual zbuffer is connected to the node, be VERY careful with the No zbuffer switch. If the Zscale value happens to be large, and you forget to set it back to some low value, the values may suddenly be interpreted as huge blur radius values that will cause processing times to explode.

Previous: Manual/Compositing Nodes Vector	Contents	*Next:* Manual/Compositing Nodes Convertors

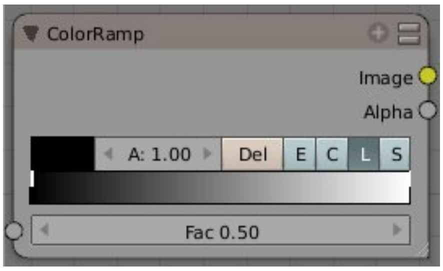
User Manual: Contents | Guidelines | Blender Version **2.44**

As the name implies, these nodes convert the colors or other propertiers of various data (e.g. images) in some way. They also split out or re−combine the different color channels that make up an image, allowing you to work on each channel independently. Various color channel arrangements are supported, including traditional RGB and HSV formats, and the newest High Definition Media Interface (HDMI) formats.

ColorRamp Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ ColorRamp



ColorRamp node

The ColorRamp Node is used for mapping values to colors with the use of a gradient. It works exactly the same way as a colorband for textures and materials, using the Factor value as a slider or index to the color ramp shown, and outputting a color value and an alpha value from the output sockets.

By default, the ColorRamp is added to the node map with two colors at opposite ends of the spectrum. A completely black black is on the left (Black as shown in the swatch with an Alpha value of 1.00) and a whitewash white is on the right. To select a color, LMB click on the thin vertical line/band within the colorband. The example picture shows the black color selected, as it is highlighted white. The settings for the color are shown above the colorband as (left to right): color swatch, Alpha setting, and interpolation type.

To change the hue of the selected color in the colorband, LMB click on the swatch, and use the popup color picker control to select a new color. Press enter to set that color.

To add colors, hold Ctrl down and Ctrl LMB click inside the gradient. Edit colors by clicking on the rectangular color swatch, which pops up a color−editing dialog. Drag the gray slider to edit A:lpha values. Note that you can use textures for masks (or to simulate the old "Emit" functionality) by connecting the alpha output to the factor input of an RGB mixer.

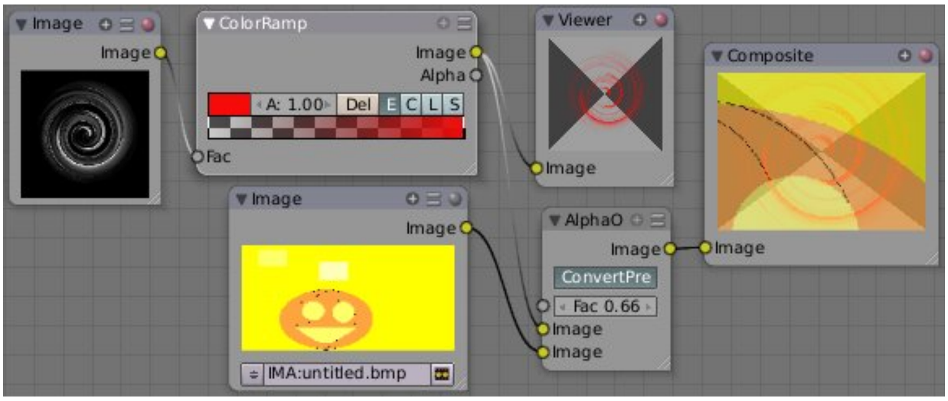
To delete a color from the colorband, select it and press the Delete button.

When using multiple colors, you can control how they transition from one to another through an interpolation mixer. Use the interpoloation buttons to control how the colors should band together: Ease, Cardinal, Linear, or Spline.

Use the A: button to define the Alpha value of the selected color for each color in the range.

Using ColorRamp to create an Alpha Mask

A powerful but often overlooked feature of the ColorRamp is to create an Alpha Mask, or a mask that is overlaid on top of another image, and, like a mask, allows some of the background to show through. The example map below shows how to use the Color Ramp node to do this:



Using the ColorRamp node to create an alpha mask

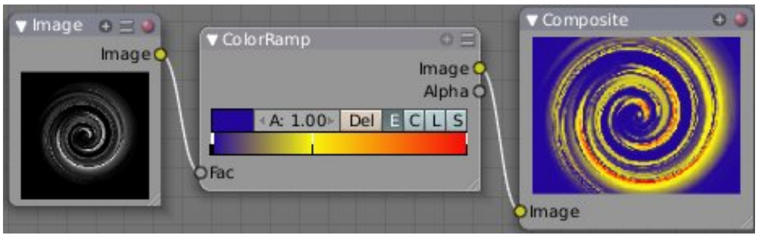
In the map above, a black and white swirl image, which is lacking an alpha channel, is fed into the ColorRamp node as a Factor. (Technically, we should have converted the image to a value using the RGB−to−BW node, buy hey, this works just as well since we are using a BW image as input.)

We have set the ColorRamp node to a purely transparent color on the left end of the spectrum, and a fully Red color on the right. As seen in the viewer, the ColorRamp node puts out a mask that is fully transparent where the image is black. Black is zero, so ColorRamp uses the 'color' at the left end of the spectrum, which we have set to transparent. The ColorRamp image is fully red and opaque where the image is white (1.00).

We verify that the output image mask is indeed transparent by overlaying it on top of a pumpkin image. For fun, we made that AlphaOver output image 0.66 transparent so that we can, in the future, overlay the image on a flashing white background to simulate a scary scene with lighting flashes.

Using ColorRamp to Colorize an Image

The real power of ColorRamp is that multiple colors can be added to the color spectrum. This example compositing map takes a boring BW image and makes it a flaming swirl!

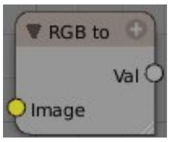


In this example, we have mapped the shades of gray in the input image to three colors, blue, yellow, and red, all fully opaque (Alpha of 1.00). Where the image is black, ColorRamp substitutes blue, the currently selected color. Where it is some shade of gray, ColorRamp chooses a corresponding color from the spectrum (bluish, yellow, to redish). Where the image is fully white, ColorRamp chooses red.

RGB to BW Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ RGB to BW



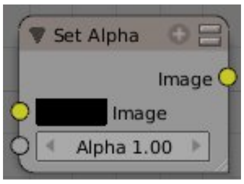
RGB to BW node

This node converts a color image to black−and−white.

Set Alpha Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Set Alpha



Set Alpha node

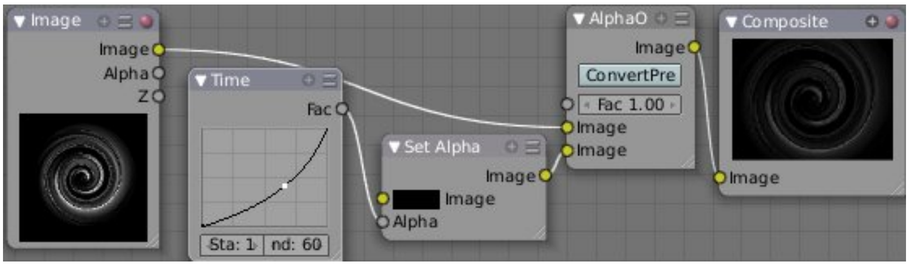
This node adds an alpha channel to a picture. Some image formats, such as JPEG, do not support an alpha channel. In order to overlay a JPEG image on top of a background, you must add an alpha channel to it using this node.

The Image input socket is optional. If an input image is not supplied, the base color shown in the swatch will be used. To change the color, LMB click the swatch and use the color−picker control to choose or specify a color you want.

The amount of Alpha (1.00 being totally opaque and 0.00 being totally tranparent) can be set for the whole picture using the input field. Additionally, the Alpha factor can be set by feeding its socket.

Using SetAlpha to Fade to Black

To transition the audience from one scene or shot to another, a common technique is to "fade to black". As its name implies, the scene fades to a black screen. You can also "fade to white' or whatever color you wish, but black is a good neutral color that is easy on the eyes and intellectually "resets" the viewer's mind. The node map below shows how to do this using the Set Alpha node.



Fade To Black

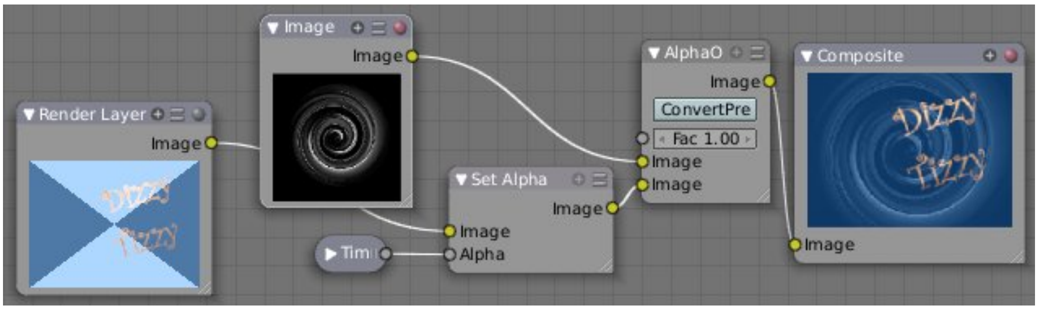
In the example above, the alpha channel of the swirl image is ignored. Instead, a time node introduces a factor from 0.00 to 1.00 over 60 frames, or about 2 seconds, to the Set Alpha node. Note that the time curve is exponentially−shaped, so that the overall blackness will fade in slowly and then accelerate toward the end. The Set Alpha node does not need an input image; instead the flat (shadeless) black color is used. The Set Alpha Node uses the input factor and color to create a black image that has an alpha set which goes from 0.00 to 1.00 over 60 frames, or completely transparent to completely opaque. Think of alpha as a multiplier for how vivid you can see that pixel. These two images are combined by our trusty AlphaOver node completely (a Factor of 1.00) to produce the composite image. The SetAlpha node will thus, depending on the frame being rendered, produce a black image that has some degree of transparency. Set up and Animate, and you have an image sequence that fades to black over a 2−second period.

No Scene information used

This example node map does not use the RenderLayer. To produce this 2 second animation, no blender scene information was used. This is an example of using Blender's powerful compositing abilities separate from its modeling and animation capabilities. (A Render Layer could be substituted for the Image layer, and the "fade−network" effect will still produce the same effect)

Using SetAlpha to Fade In a Title

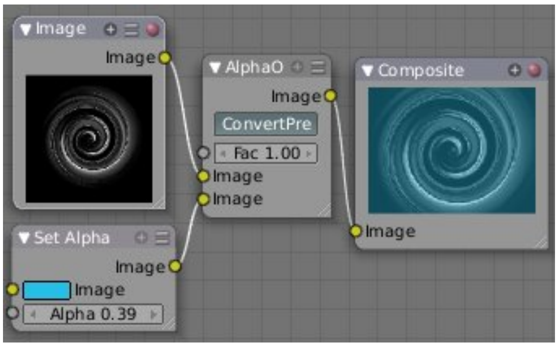
To introduce your animation, you will want to present the title of your animation over a background. You can have the title fly in, or fade it in. To fade it in, use the SetAlpha node with the Time node as shown below.



Using Set Alpha to Fade in a Title

In the above example, a Time curve provides the Alpha value to the input socket. The current RenderLayer, which has the title in view, provides the image. As before, the trusty AlphaOver node mixes (using the alpha values) the background swirl and the alphaed title to produce the composite image. Notice the ConvertPre−Multiply button is NOT enabled; this produces a composite where the title lets the background image show through where even the background image is transparent, allowing you to layer images on top of one another.

Using SetAlpha to Colorize a BW Image



Using Set Alpha to Colorize an Image

In the example above, notice how the blue tinge of the render input colors the swirl. You can use the Set Alpha node's color swatch with this kind of node map to add a consistent color to a BW image.

In the example map to the right, use the Alpha value of the SetAlpha node to give a desired degree of colorization. Thread the input image and the Set Alpha node into an AlphaOver node to colorize any black and white image in this manner. Note the ConvertPre−Multiply button is enabled, which tells the AlphaOver node not to multiply the alpha values of the two images together.

ID Mask Node

Panel: Node Editor Node Composition

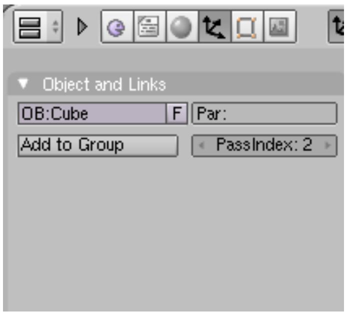
Menu: Shift A â†’ Convertors â†’ ID Mask

ID Mask node

This node will use the Object Index pass (see RenderLayers) to produce an anti−aliased alpha mask for the object index specified. The mask is opaque where the object is, and transparent where the object isn't. If the object is partially transparent, the alpha mask matches the object's transparency. This post−process function fills in the jaggies with interpolated values.

Object Index

Object indices are only output from a RenderLayers node or stored in a multilayer OpenEXR format image.

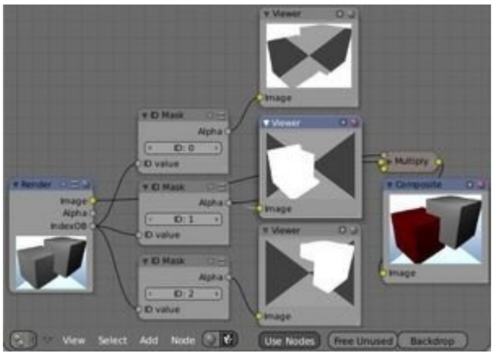


Setting an Object Index

You can specify, for any of the objects in your scene, an Object Index as shown the right (the currently select object has an index of 2). When rendered, if Object Index passes are enabled, its index will be 2, and setting the ID Mask node to 2 will show where that object is in the scene.

This node is extreeeemely well suited to removing the aliases shown as output from the Defocus node or DOF noodles caused by some objects being close to camera against objects far away.

Example



Example

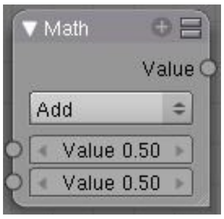
In this example, the left rear red cube is assigned PassIndex 1, and the right cube PassIndex 2. Where the two cubes intersect, there is going to be noticeable pixelation (jaggies) because they come together at a sharp angle and are different colors. Using the mask from object 1, which is smoothed (anti−aliased) at the edges, we use a Mix node set on Multiply to multiply the smoothed edges against the image, thus removing those nasty (Mick) Jaggies. Thus, being smoothed out, the Rolling Stones gather no moss. (I really hope you get that obscure reference :)

Note that the mask returns white where the object is fully visible to the camera(not behind anything else) and black for the part of the object that is partially or totally obscured by a fully or partially opaque object in front of it. If something else is in front of it, even if that thing is partially transparent and you can see the object in a render, the mask will not reflect that partially obscured part.

Math Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Math



Math node

This node performs the selected math operation on an image or buffer. All common math functions are supported. If only an image is fed to one Value socket, the math function will apply the other Value consistently to every pixel in producing the output Value. Select the math function by clicking the up−down selector where the "Add" selection is shown.

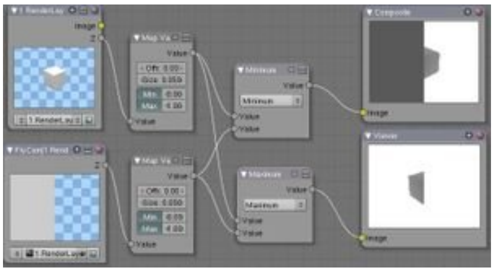
The trig functions of Sine, Cosine, Tangent use only the top socket and accept values in radians between 0 and 2*pi for one complete cycle.

V.: 2.44

Known bug: the Top socket must get the image if the bottom socket is left as a value.

Examples

Manual Z−Mask

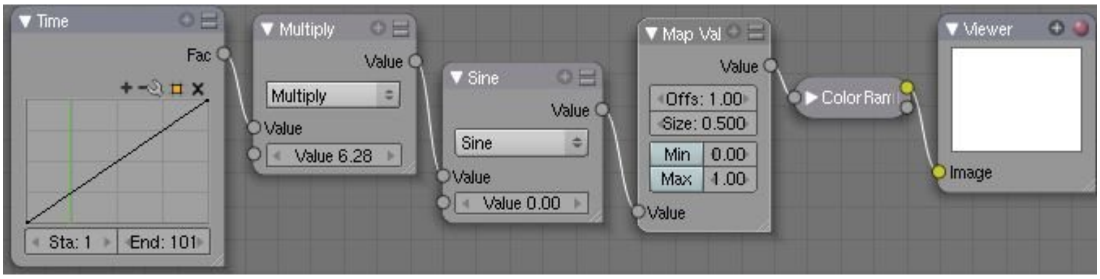


Example

This example has one scene input by the top RenderLayer node, which has a cube that is about 10 BU from the camera. The bottom RenderLayer node inputs a scene (FlyCam) with a plane that covers the left half of the view and is 7 BU from the camera. Both are fed through their respective Map Value nodes to divide the Z buffer by 20 (multiply by .05, as shown in the Size field) and clamped to be a Min/Max of 0.0/1.0 respectively.

For the Minimum function, the node selects those Z values where the corresponding pixel is closer to the camera; so it chooses the Z values for the plane and part of the cube. The background has an infinite Z value, so it is clamped to 1.0 (shown as white). In the maximum example, the Z values of the cube are greater than the plane, so they are chosen for the left side, but the plane (FlyCam) Renderlayer's Z are infinite (mapped to 1.0) for the right side, so they are chosen.

Using Sine Function to Pulsate

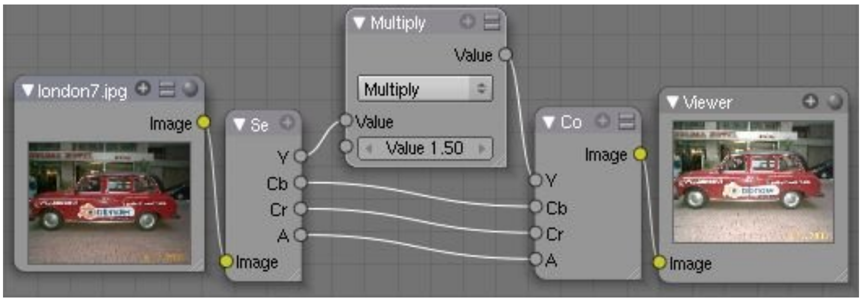


This example has a Time node putting out a linear sequence from 0 to 1 over the course of 101 frames. The green vertical line in the curve widget shows that frame 25 is being put out, or a value of .25. That value is multiplied by 2*pi and converted to 1.0 by the Sine function, since we all know that Sine(2*pi/4)=Sine(pi/2)=+1.0.

Since the Sine function can put out values between −1.0 and 1.0, the Map Value node scales that to 0.0 to 1.0 by taking the input (−1 to 1), adding 1 (making 0 to 2), and multiplying the result by one half (thus scaling the output between 0 and 1). The default ColorRamp converts those values to a grayscale. Thus, medium gray corresponds to a 0.0 output by the sine, black to −1.0, and white to 1.0. As you can see, Sine(pi/2)=1.0. Like having your own visual color calculator! Animating this noodle provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move an scene in/out of focus. Alter a color channel value to make a color "pulse".

Brightening/Scaling a Channel



This example has a Multiply node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a Map Value node with Min() and Max() enabled to clamp the output to valid values. With this approach you could use a logarithmic function to make a high−dynamic range image. For this particular example, there is also a Brighten/Contrast node that might give simpler control over brightness.

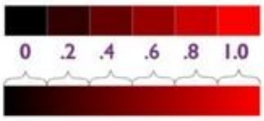
Quantize/Restrict Color Selection

In this example, we want to restrict the color output to only 256 possible values. Possible use of this is to see what the image will look like on an 8−bit cell phone display. To do this, we want to restrict the R, G and B values of any pixel to be one of a certain value, such that when they are combined, will not result in more than 256 possible values. The number of possible values of an output is the number of channel values multiplied by each other, or Q = R * G * B.

Since there are 3 channels and 256 values, we have some flexibility how to quantize each channel, since there are a lot of combinations of R*G*B that would equal 256. For example, if {R,G,B} = {4,4,16}, then 4 * 4 * 16 = 256. Also, {6,6,7} would give 252 possible values. The difference in appearance between {4,4,16} and

{6,6,7} is that the first set (4,4,16} would have fewer shades of red and green, but lots of shades of blue. The set {6,6,7} would have a more even distribution of colors. To get better image quality with fewer color values, give more possible values to the predominant colors in the image.

Theory



Two Approaches to Quantizing to 6 values

To accomplish this quantization of an image to 256 possible values, lets use the set {6,6,7}. To split up a continuous range of values between 0 and 1 (the full Red spectrum) into 6 values, we need to construct an algorithm or function that takes any input value but only puts out 6 possible values, as illustrated by the image to the right. We want to include 0 as true black, with five other colors in between. The approach shown produces {0,.2,.4,.6,.8,1}. Dividing 1.0 by 5 equals .2, which tells us how far apart each quantified value is from the other.

So, to get good even shading, we want to take values that are 0.16 or less and map them to 0.0; values between 0.16 and 0.33 get fixed to 0.2; colorband values between 0.33 and 0.5 get quantized to 0.4, and so on up to values between 0.83 and 1.0 get mapped to 1.0.

Function f(x)

An algebraic function is made up of primitive mathematical operations (add, subtract, multiply, sine, cosine, etc) that operate on an input value to provide a desired output value.

Input		Function				Output	
8-bit	R value	*n	- ½	round()	/(n-1)	R value	8-bit
0	0.00	0.00	-0.50	0	0	0	0
13	0.05	0.30	-0.20	0	0	0	0
26	0.10	0.60	0.10	0	0	0	0
38	0.15	0.90	0.40	0	0	0	0
51	0.20	1.20	0.70	1	0.2	0.2	51
64	0.25	1.50	1.00	1	0.2	0.2	51
77	0.30	1.80	1.30	1	0.2	0.2	51
89	0.35	2.10	1.60	2	0.4	0.4	102
102	0.40	2.40	1.90	2	0.4	0.4	102
115	0.45	2.70	2.20	2	0.4	0.4	102
128	0.50	3.00	2.50	2	0.4	0.4	102
140	0.55	3.30	2.80	3	0.6	0.6	153
153	0.60	3.60	3.10	3	0.6	0.6	153
166	0.65	3.90	3.40	3	0.6	0.6	153
179	0.70	4.20	3.70	4	0.8	0.8	204
191	0.75	4.50	4.00	4	0.8	0.8	204
204	0.80	4.80	4.30	4	0.8	0.8	204
217	0.85	5.10	4.60	5	1	1	255
230	0.90	5.40	4.90	5	1	1	255
242	0.95	5.70	5.20	5	1	1	255
255	1.00	6.00	5.50	5	1	1	255

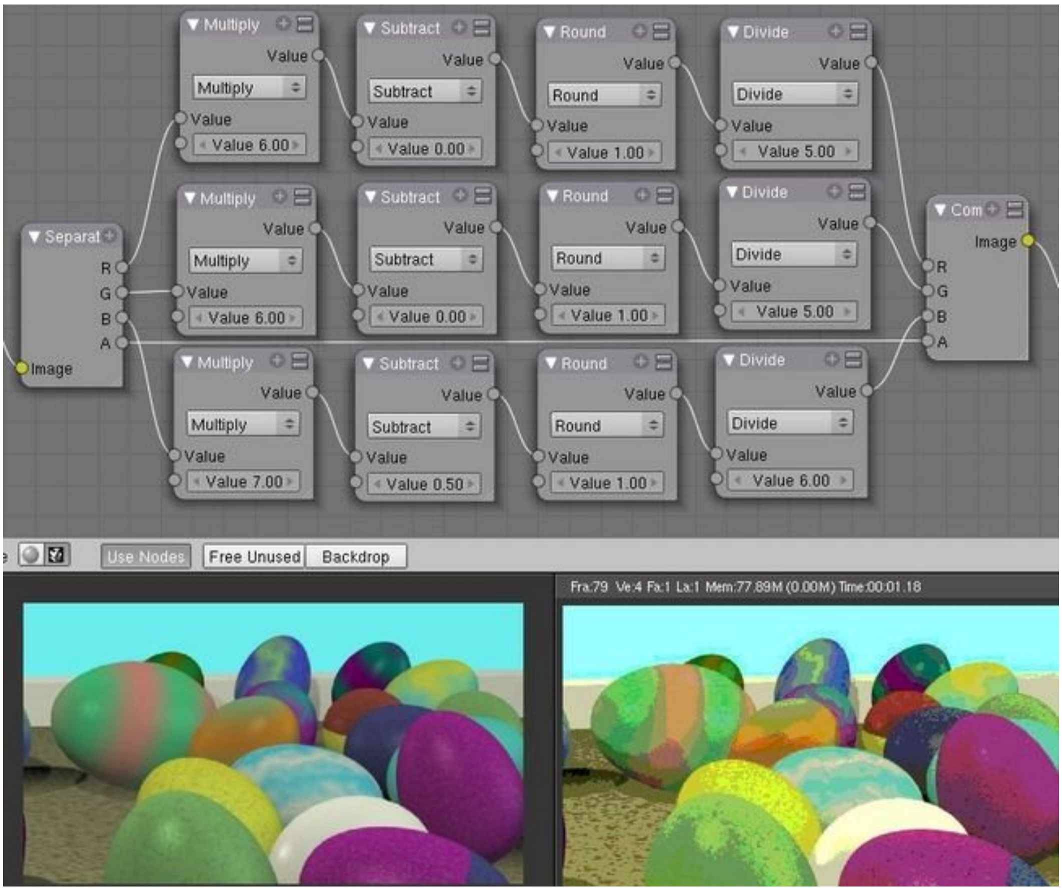
The theory behind this function is scaled truncation. Let us suppose we want a math function that takes in a range of values between 0 and 1, such as .552, but only outputs a value of 0.0, 0.2, 0.4, etc. We can imagine then that we need to get that range 0 to 1 powered up to something 0 to 6 so that we can chop off and make it a whole number. So, with six divisions, how can we do that? The answer is we multiply the range by 6. The output of that first math multiply node is a range of values between 0 and 6. To get even divisions, because we are using the rounding function (see documenation above), we want any number plus or minus around a whole number will get rounded to that number. So, we subtract a half, which shifts everything over. The Round() function then makes that range 0 to 5. We then divide by 5 to get back a range of numbers between 0 and 1 which can then be combined back with the other color channels. Thus, you get the the function

f(x,n)=round[x*n−1/2] /(n−1)

where n is the number of possible output values, and x is the input pixel color and f(x,n) is the output value. There's only one slight problem, and that is for the value exactly equal to 1, the formula result is 1.2, which is an invalid value. This is because the round function is actually a roundup function, and exactly 5.5 is rounded up to 6. So, by subtracting .501, we compensate and thus 5.499 is rounded to 5. At the other end of the spectrum, pure black, or 0, when .501 subtracted, rounds up to 0 since the Round() function does not return a negative number.

Sometimes using a spreadsheet can help you figure out how to put these nodes together to get the result that you want. Stepping you through the formula for n=6 and x=0.70, locate the line on the spreadsheet that has the 8−bit value 179 and R value 0.7. Multiplying by 6 gives 4.2. Subtracting 1/2 gives 3.7, which rounds up to 4. 4 divided by 5 = .8. Thus, f(0.7, 6) = 0.8 or an 8−bit value of 204. You can see that this same 8−bit value is output for a range of input values. Yeah! Geeks Rule! This is how you program Blender to do compositing based on Algebra. Thank a Teacher if you understand this.

Reality



To implement this function in Blender, consider the noodle above. First, feed the image to the Separate RGB node. For the Red channel, we string the math nodes into a function that takes each red color, multiplies (scales) it up by the desired number of divisions (6), offsets it by 0.5, rounds the value to the nearest whole number, and then divides the image pixel color by 5. So, the transformation is {0..1} becomes {0..6}, subtracting centers the medians to {−0.5...5.5} and the rounding to the nearest whole number produces {0,1,2,3,4,5} since the function rounds down, and then dividing by five results in six values {0.0,0.2,0.4,0.6,0.8,1.0}.

The result is that the output value can only be one of a certain set of values, stair−stepped because of the rounding function of the math node noodle. Copying this one channel to operate on Green and Blue gives the noodle below. To get the 6:6:7, we set the three multiply nodes to {6,6,7} and the divide nodes to {5,5,6}.

If you make this into a node group, you can easily re−use this setup from project to project. When you do, consider using a math node to drive the different values that you would have to otherwise set manually, just to error−proof your work.

Summary

Normally, an output render consists of 32− or 24−bit color depth, and each pixel can be one of millions of possible colors. This noodle example takes each of the Red, Green and Blue channels and normalizes them to one of a few values. When all three channels are combined back together, each color can only be one of 256 possible values.

While this example uses the Separate/Combine RGB to create distinct colors, other Separate/Combine nodes can be used as well. If using the YUV values, remember that U and V vary between −0.5 and +0.5, so you will have to first add on a half to bring the range between 0 and 1, and then after dividing, subtract a half to bring in back into standard range.

The JPG or PNG image format will store each of the colors according to their image standard for color depth (e.g. JPG is 24−bit), but the image will be very very small, since reducing color depth and quantizing colors is essentially what the JPEG compression algorithm accomplishes.

You do not have to reduce the color depth of each channel evenly. For example, if blue was the dominant color in an image, to preserve image quality, you could reduce Red to 2 values, Green to 4, and let the blue take on 256/(2*4) or 32 values. If using the HSV, you could reduce the Saturation and Value to 2 values (0 or 1.0) by Multiply by 2 and Divide by 2, and restrict the Hue to 64 possible values.

You can use this noodle to quantize any channel; alpha, speed (vector), z−values, and so forth.

Combine/Separate Nodes

All of these node do essentially the same thing: they split out an image into (or recombine an image from) its composite color channels. Each format supports the Alpha (transparency) channel. The standard way of representing color in an image is called a *color space*. There are several color spaces supported:

- RGB: Red−Green−Blue traditional primary colors, also broadcast directly to most computer monitors
- HSV: Three values, often considered as more intuitive than the RGB system (nearly only used on computers):
    - Hue: the **Hue** of the color (in some way, choose a 'color' of the rainbow);
    - Saturation: the **quantity** of hue in the color (from desaturate − shade of gray − to saturate − brighter colors);

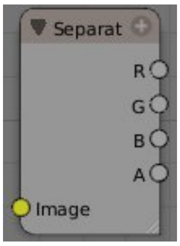
- Value: the **luminosity** of the color (from 'no light' − black − to 'full light' − 'full' color, or white if Saturation is 0.0).
- YUV: Luminance−Chrominance standard used in broadcasting analog PAL (European) video.
- YCbCr: Luminance−ChannelBlue−ChannelRed Component video for digital broadcast use, whose standards have been updated for HDTV and commonly referred to as the HDMI format for component video.

See the global wikipedia for more information on color spaces.

Separate/Combine RGBA Node

Panel: Node Editor Node Composition

Menu: Shift A â†' Convertors â†' Separate RGBA

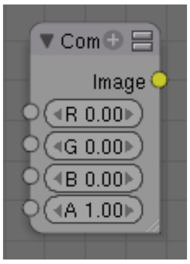


Separate RGBA node

This node separates an image into its red, green, blue and alpha channels. There's a socket for each channel on the right.

Panel: Node Editor Node Composition

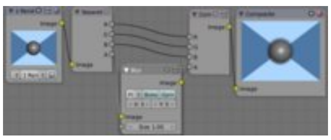
Menu: Shift A â†' Convertors â†' Combine RGBA



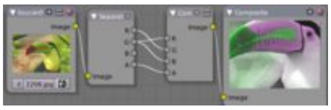
Combine RGBAnode

This node combines separate input images as each color and alpha channel, producing a composite image. You use this node combine the channels after working on each color channel separately.

Examples



In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like anti−aliasing, but in a three−dimensional sense. Use this noodle when adding CG elements to live action to remove any hard edges. Animating this effect over a broader scale will make the object appear to "phase" in and out, as a "out−of−phase" time−traveling sync effect.

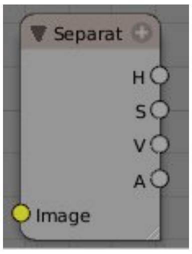


In this fun little noodle we make all the reds become green, and all the green both Red and Blue, and remove Blue from the image completely. Very cute. Very fun.

Separate/Combine HSVA Nodes

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Separate HSVA



Separate HSVA node

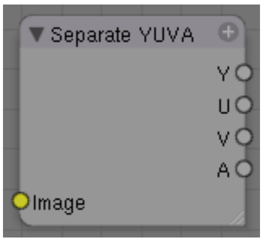
This node separates an image into image maps for the hue, saturation, value and alpha channels.

Use and manipulate the separated channels for different purposes; i.e. to achieve some compositing/color adjustment result. For example, you could expand the Value channel (by using the multiply node) to make all the colors brighter. You could make an image more relaxed by diminishing (via the divide or map value node) the Saturation channel. You could isolate a specific range of colors (by clipping the Hue channel via the Colorramp node) and change their color (by the Add/Subtract mix node).

Separate/Combine YUVA Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Separate YUVA

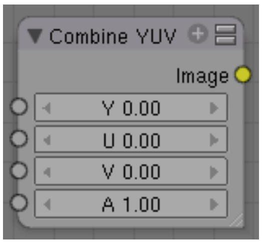


Separate YUVA node

This node converts an RGBA image to YUVA color space, then splits each channel out to its own output so that they can be manipulated independently. Note that U and V values range from −0.5 to +0.5.

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Combine YUVA



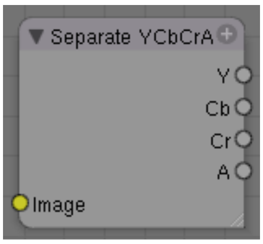
Combine YUVA node

Combines the channels back into a composite image. If you do not connect any input socket, you can set a default value for the whole image for that channel using the numeric controls shown.

Separate/Combine YCbCrA Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Separate YCbCrA



Separate YCbCrA node

This node converts an RGBA image to YCbCrA color space, then splits each channel out to its own output so that they can be manipulated independently:

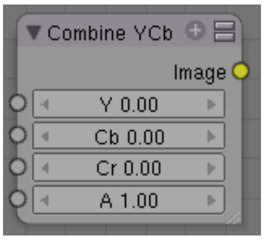
- Y: Luminance, 0=black, 1=white

- Cb: Chrominance Blue, 0=Blue, 1=Yellow
- Cr: Chrominance Red, 0=Red, 1=Yellow

Note: If running these channels through a ColorRamp to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives high−contrast effect.

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Combine YCbCrA



Combine YCbCrA node

So, I kinda think you get the idea, and I was trying to think of some other creative way to write down the same thing, but I can't. So, you'll have to figure this node out on your own.

Previous: Manual/Compositing Nodes Filters	Contents	*Next:* Manual/Compositing Nodes Mattes

User Manual: Contents | Guidelines | Blender Version **2.43**

These nodes give you the essential tools for working with blue−screen or green−screen footage, where live action is shot in front of a blue or green backdrop for replacement by a matte painting or virtual background.

In general, hook up these nodes to a viewer, set your UV/Image Editor to show the viewer node, and play with the sliders in real−time using a sample image from the footage, to get the settings right. In some cases, small adjustments can eliminate artifacts or foreground image degredation. For example, taking out too much green can result in foreground actors looking 'flat' or blueish/purplish.

You can and should chain these nodes together, refining your color correction in successive refinements, using each node's strengths to operate on the previous node's output. There is no "one stop shopping" or one "does−it−all" node; they work best in combination.

Usually, green screen is shot in a stage with consistent lighting from shot to shot, so the same settings will work across multiple shots of raw footage. Footage shot outside under varying lighting conditions (and wind blowing the background) will complicate matters and mandate lower falloff values.

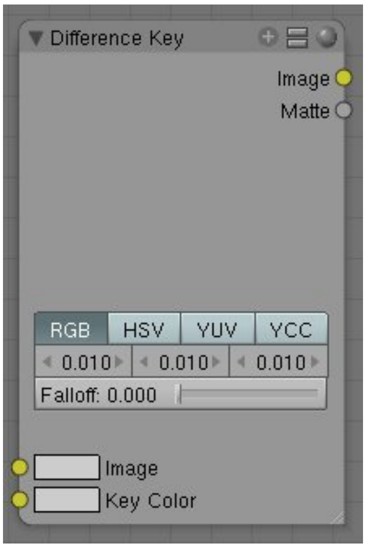
Garbage Matte

Garbage matte is not a node, but a technique where the foreground is outlined using a closed curve (bezier or nurbs). Only the area within the curve is processed using these matte nodes; everything else is garbage and thus discarded.

Difference Key Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Mattes â†’ Difference Key



Difference Key node

The difference key node determines how different each channel is from the given key in the selected color space. If the differences are below a user defined threshold then the pixel is considered transparent. Difference matting does not rely on a certain background color, but can have less than optimal results if there is a

significant amount of background color in the foreground object.

There are two inputs to this node.

- The first is an input Image that is to be keyed.
- The Key Color can be input as an RGB value or selected using the color picker by clicking on the Key Color box to bring up the color dialog, then clicking on the eye dropper tool and selecting a color.

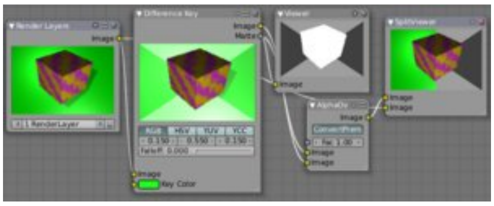
The selectable color spaces are RGB (default), HSV, YUV, and YCbCr.

You can adjust the tolerance of each color in the colorspace individually so that you can have more red variance or blue variance in what you would allow to be transparent. I find that about 0.15 (or 15%) is plenty of variance if the background is evenly lit. Any more unevenness and you risk cutting into the foreground image.

When the Falloff value is high, pixels that are close to the Key Color are more transparent than pixels that are not as close to the Key Color (but still considered close enough to be keyed). When the Falloff value is low, it does not matter how close the pixel color (Image) is to the Key Color, it is transparent.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

Simple Example

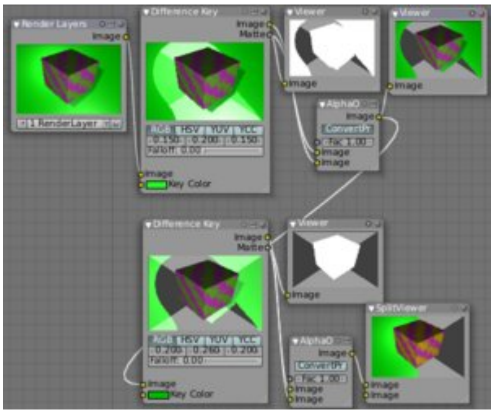


Using the Difference Key Node

In the example to the right (click to expand), we have a purple cube with yellow marbeling in front of a very unevenly lit green screen. We start building our noodle by threading the image to a difference key, and using the eyedropper, pick a key color very close to the edge of the cube, around where the halo is at the corner on the left−hand side; a fairly bright green. We thread two viewers from the output sockets so we can see what (if anything) the node is doing. We add an AlphaOver node, threading the Matte to the **TOP** socket and the image to the **BOTTOM** socket. Very Important, because 0 time blue is not the same as blue times zero. You always want your mask to go to the top socket of the AlphaOver. Premultiply is set and a full multiply is on so that we completely remove the green. In this example, we thread the output of the alphaover to a SplitViewer node so we can compare our results; the original is threaded to the bottom input of the SplitViewer, so that original is on the left, processed is on the right.

We set our variance to .15, and see what we get. What we get (not shown) is a matte that masks around the cube, but not on the right and around the edges where the green is darker; that shade it is too far away from our key color. So, since it is the green that is varying that we want to remove, we increase the Green variation to 1.00 (not shown). Whoa! All the Green disappears (all green within a 100% variation of our green key color is *all* the green), along with the top of the box! Not good. So, we start decreasing the green until we settle on 55% (shown).

Chaining Example



Chaining Difference Key Nodes

We pay out the wazoo for our highly talented (and egotistical I might add) Mr. Cube to come into the studio and do a few takes. We told him NOT to wear a green tie, but when we look at our footage, lo and behold, there he is with a green striped tie on. When we use our simple noodle, the green stripes on his tie go alpha, and the beach background shows through. So, we call him up and, too late, he's on his way back to Santa Monica and it wasn't in his contract and it wasn't his fault, after all, we're supposed to have all this fancy postpro software yada yada and he hangs up. Geez, these actors.

So, we chain two Difference Key nodes as shown to the right, and problem solved. What we did was lower the variation percentage on the first to remove some of the green, then threaded that to a second (lower) difference key, where we sampled the green more toward the shadow side and outside edge. By keeping both variations low, none of the green in his tie is affected; that shade is outside the key's +/− variation tolerances.

Chroma Key Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Mattes â†’ Chroma Key

Chroma Key node

The Chroma Key node determines if a pixel is foreground or background (and thereby should be transparent) based on its chroma values. This is useful for compositing images that have been shot in front of a green or blue screen.

There is one input to this node, the Image that is to be keyed.

Control this node using:

Green / Blue buttons
Basic selection of what color the background is supposed to be.

Cb Slope and Cr Slope (chroma channel) sliders
Determines how quickly the processed pixel values go from background to foreground, much like falloff.

Cb Pos and Cr Pos sliders
Determines where the processing transition point for foreground and background is in the respective channel.

Threshold
Determines if additional detail is added to the pixel if it is transparent. This is useful for pulling shadows from an image even if they are in the green screen area.

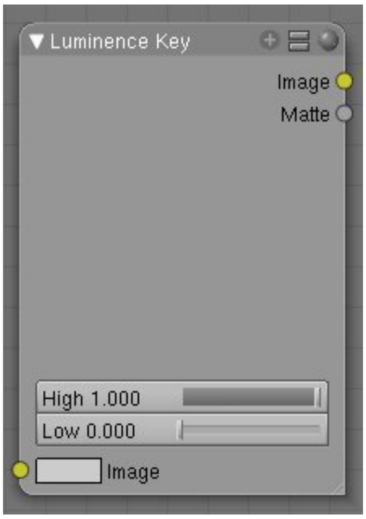
Alpha threshold
The setting that determines the tolerance of pixels that should be considered transparent after they have been processed. A low value means that only pixels that are considered totally transparent will be transparent, a high value means that pixels that are mostly transparent will be considered transparent.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

Luminance Key Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Mattes â†’ Luma Key



Luminance Key node

The Luminance Key node determines background objects from foreground objects by the difference in the luminance (brightness) levels. For example, this is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.

There is one input to this node, the Image that is to be keyed.

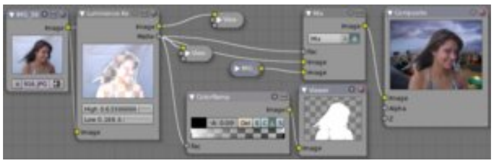
Control this node using:

- The High value selector determines the lowest values that are considered foreground. (which is supposed to be − relatively − light: from this value to 1.0).
- The Low value selector determines the hightes values that are considered to be background objects. (which is supposed to be − relatively − dark: from 0.0 to this value).

It is possible to have a separation between the two values to allow for a gradient of transparency between foreground and background objects.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

Example



Using Luma Key...with a twist

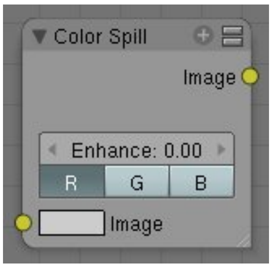
For this example, let's throw you a ringer. Here, the model was shot against a *white* background. Using the Luminance Key node, we get a matte out where the background is white, and the model is black; the opposite of what we want. If we wanted to use the matte, we have to switch the white and the black. How to do this? ColorRamp to the rescue − we set the left color White Alpha 1.0, and the right color to be Black Alpha 0.0. Thus, when the Colorramp gets in black, it spits out white, and vice versa. The reversed mask is shown; her white outline is useable as an alpha mask now.

Now to mix, we don't really need the AlphaOver node; we can just use the mask as our Factor input. In this kinda weird case, we can use the matte directly; we just switch the input nodes. As you can see, since the matte is white (1.0) where we don't want to use the model picture, we feed the background photo to the bottom socket (recall the mix node uses the top socket where the factor is 0.0, and the bottom socket where the factor is 1.0). Feeding our original photo into the top socket means it will be used where the Luminance Key node has spit out Black. Voila, our model is teleported from Atlanta to aboard a cruise ship docked in Miami.

Color Spill Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Mattes â†’ Color Spill



Color Spill node

The Color Spill node reduces one of the RGB channels so that it is not greater than any of the others. This is common when compositing images that were shot in front of a green or blue screen. In some cases, if the foreground object is reflective, it will show the green or blue color; that color has "spilled" onto the foreground object. If there is light from the side or back, and the foreground actor is wearing white, it is possible to get "spill" green (or blue) light from the background onto the foreground objects, coloring them with a tinge of green or blue. To remove the green (or blue) light, you use this fancy node.

There is one input to this node, the Image to be processed.

The Enhance slider allows you to reduce the selected channel's input to the image greater than the color spill algorithm normally allows. This is useful for exceptionally high amounts of color spill.

The outputs of this node are the image with the corrected channels.

Channel Key Node

V.: 2.44

Panel: Node Editor Node Composition

Menu: Shift A â†’ Mattes â†’ Channel Key

Channel Key node

The Channel Key node determines background objects from foreground objects by the difference in the selected channel's levels. For example in YUV color space, this is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.

There is one input to this node, the Image that is to be keyed.

Control this node using:

- Color Space buttons selects what color space the channels will represent.
- Channel buttons selects the channel to use to determine the matte.
- High value selector determines the lowest values that are considered foreground. (which is supposed to be − relatively − hight values: from this value to 1.0).
- Low value selector determines the highest values that are considered to be background objects. (which is supposed to be − relatively − low values: from 0.0 to this value).

It is possible to have a seperation between the two values to allow for a gradient of transparency between foreground and background objects.

The outputs of this node are the Image with an alpha channel adjusted for the keyed selection and a black and white Matte (i.e the alpha mask).

Previous: Manual/Compositing Nodes Convertors	Contents	*Next:* Manual/Compositing Nodes Distort

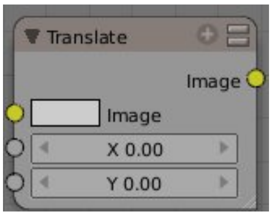
User Manual: Contents | Guidelines | Blender Version **2.43**

These nodes distort the image in some fashion, operating either uniformly on the image, or by using a mask to vary the effect over the image.

Translate Node

Panel: Node Editor Node Composition

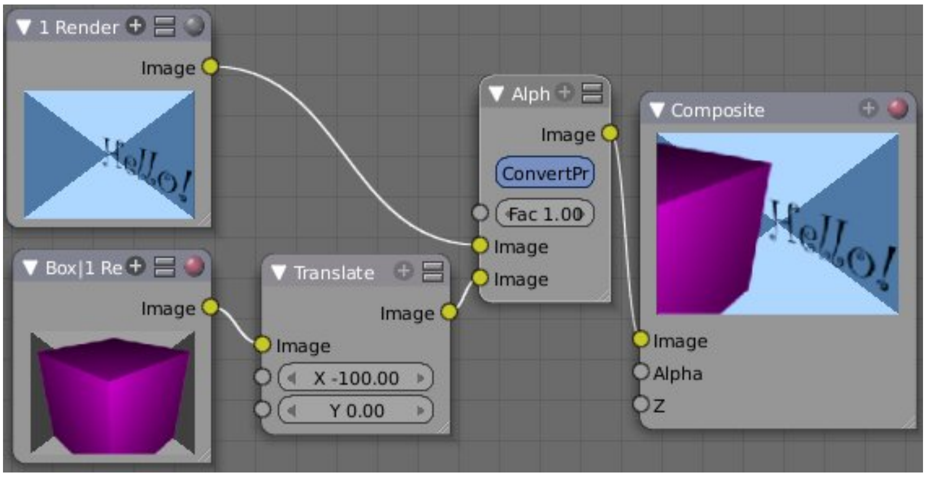
Menu: Shift A â†’ Convertors â†’ Translate



Translate node

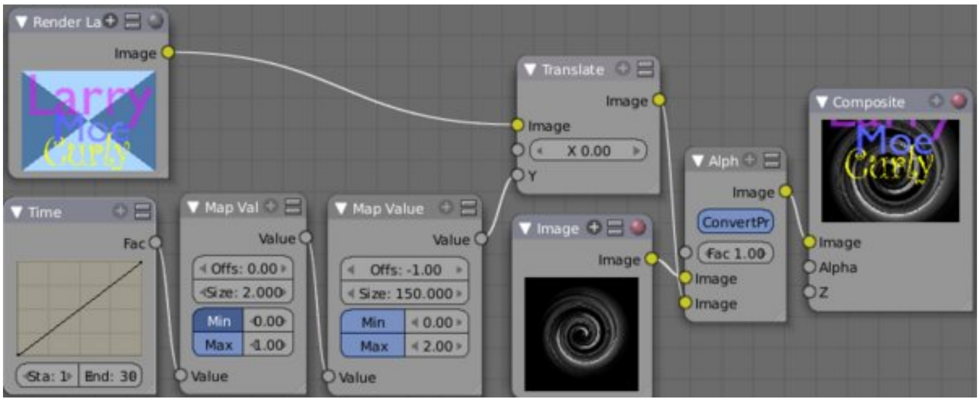
The translate node translates (moves) an image by the specified amounts in the X and Y directions. X and Y are in pixels, and can be positive or negative. To shift an image up and to the left, for example, you would specify a negative X offset and a positive Y.

Use this node to position an image into the final composite image. Usually, the output of this node is routed to an AlphaOver or Mix node to mix it with a base image. In the example below, the RenderLayer input from one scene (box) is translated over to the left (a negative X translation) and alphaovered with a Hello scene RenderLayer input



Example: Using the Translate Node to Roll Credits

At the end of your fantastic animation, you'll want to give credit where credit is due. This is called rolling the credits and you see the names of everyone involved scroll up over a background image or sequence. The mini−map below shows an example of how to roll credits using the Translate node.



Using the Translate Node to Roll Credits

In this node map, the RenderLayer input has a list of the people involved and is 150 pixels high; it is the image input into the Translate Node. The Y value (vertical) offset of the Translate node comes from a scaled time factor that varies from −150 to 150 over 30 frames. The Translated image is overlaid on top of a background swirl image. So, over the course of 30 frames, the Translate node shifts the image from down by 150 pixels (off the bottom of the screen), up through and overlaid on top of the swirl, and finally off the screen to the top. These frames are generated when the Render Animation buttons are set and Anim is pressed. Right now, frame 21 is showing Moe and Curly, and Larry has scrolled off the screen.

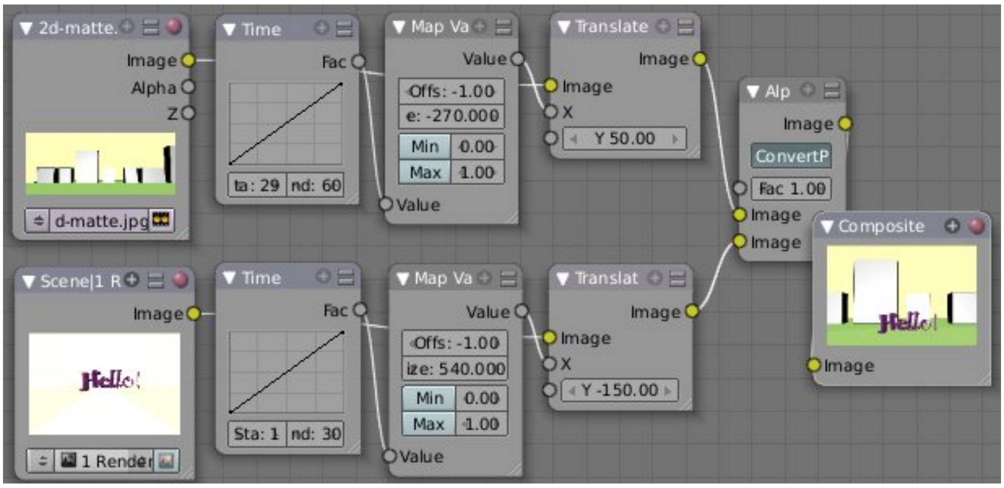
Alpha channel

Be sure to save your credits image in a format that supports an alpha channel, such as PNG, so that the AlphaOver node can overlay it on the background and let the background show through.

You *could* have parented and animated all of the text to roll up past your camera, but rendering would have taken forever. Using the translate node is much faster to composite, and more flexible. To add someone to the list, simply change the credits image and re−animate. Since it is simple image manipulation, Blender is blazingly fast at regenerating your credits. Similarly, changing the background is rock simple as well; simply load up a different background image and re−Animate.

Example: Moving a Matte

In some 2D and 3D animations and movies, a matte painting is used as the background. In most scenes it is still, however you can easily move it using the Translate node. Mattes are huge; the example used below is actually 1440x600 pixels, even though the scene being rendered is for TV. This oversizing gives us room to move the matte around. The example noodle below introduces a dancing "Hello!" from stage right in frames 1−30. As the "Hello" reaches center stage, we fake a camera move by moving the matte to the left, which makes it look like the "Hello!" is still moving as the camera moves with it.



Moving a Matte in back of a moving Animation (frame 60)

Use the Map Value node to scale the X (left to right) offsets that feed the Translate node. Note that offsets are used to position the dancing "Hello!" down to look like it is walking along the street (in the render scene, it is centered on camera and just dances in place). The matte is adjusted up to fake a camera height of an observer, bringing the horizon up.

Example: Shake, Rattle and Roll

A real world effect is the shaking of the camera. BOOM! We expect to feel the impact and see it rock our world. In our virtual CG world, though, we are in the vaccuum of space. So, we have to fake a camera being shook. There are two points in the development cycle to do this: at *Render−Scene* time, and at *Composite* time.

At *Render−Scene* time, the modeler would introduce an Ipo curve and keyframes that rotate and/or move the camera for a short amount of time. The advantage to render−scene shake is that the artist handles it and the editor does not have to worry about it; one less thing to do thank goodness. The disadvantage is that the artist may only be modeling the actors, and not the background scenery, props, or matte, so any shake they introduce does not transfer to the set, props, or backdrop. Therefore, it is best to introduce camera shake after all scenes have been rendered and assembled and when they are being composited.

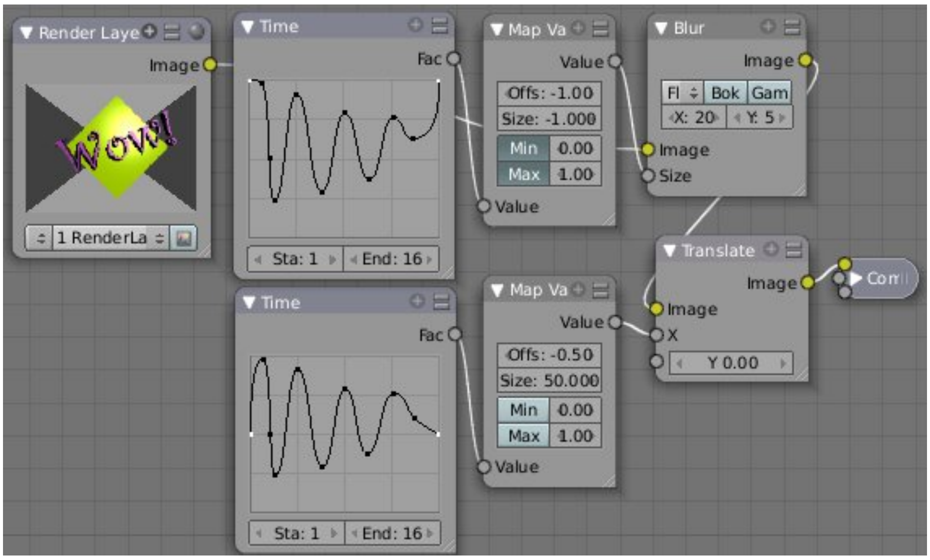
There are two aspects to being bumped, or tripping over the tripod, or having an explosion go off next to you, or an airplane have a near miss as it flies by, or throwing up on a long sea voyage, or surviving an earthquake: *camera motion* and *image blur* (I know you were thinking expletives and changing your underpants, but this is about compositing).

Camera Motion happens because the camera physically gets moved; but its mass and its tripod also acts as a dampening device, softening out and absorbing the initial bump. The cameraman also acts as a dampener, and also as a corrector, trying to get the camera back to where it was pointed originally.

There can be quite a delay between the shock and the correction; for example, a lone actor/cameraman may trip on the tripod coming out from behind the camera, come into frame, realize the camera is off, and then come back to correct it. It all depends on the artistic effect and story you want to convey.

The *image blur* comes into play because the shake happens so rapidly that the image is blurred in the direction of the shake. However, the blur is more when the camera is being pushed back into position, and less when the camera is at the extreme of its deflection, since it is decellerating at the apex of its movement. Like motion, blur is the most during the initial shock, and less as things slow down and get under control. Also, the camera may go out of focus and come back into focus at the end of the shake.

To use Blender nodes to mimic Camera Motion, use the noodle shown below. The noodle has a Blur group on top that feeds a Translate group below it.



SFX: Camera Shake

In the above example, we use a Time curve that mimics the intensity and duration of a typical BOOM!. In this case, both curves have four peaks within a 16−frame period to mimic a BOOM! (in fact one curve was constructed and then duplicated to make the other, to ensure that the bulk of both curves was exactly the same). Notice how the curve dampens (decreases in magnitude as time progresses) as discussed above. Notice how the curve slows down (increasing period) to mimic the cameraman getting it back under control. Notice that the curve is sinusoidal to mimic over−correction and vibration.

BOOM! to the Left: The translate curve starts at 0.5. Maximum deflection up is fully a half, yet down is only a quarter. This offset mimics a BOOM! off to our left, since the camera shakes more to the right, away from the BOOM!

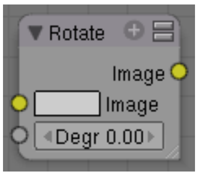
Motion and Blur are the same but different: Notice that the two curves are identical except for the highlighted start and end dots; we want zero blur and zero offsets before and after the shake, but minimum blur when there is maximum translate. The two Map Value node settings are different to achieve this; the math is left to the reader.

Use this Blender noodle to mimic camera shake. The amount of shake is set by the Size values, and the blur should be proportional to the amount and direction of motion (predominantly X in this example). Use the Time start and end to vary the duration of the shake; ten seconds for an earthquake, one minute for a ship rolling in the seas, a half second as an F−14 flies by and takes your ear off. *Author's note: I noticed cool camera shakes while watching the Halo 3 previews.*

Rotate Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Convertors â†’ Rotate



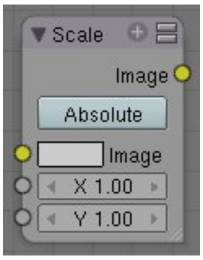
Rotate node

This node rotates an image. Positive values rotate clockwise and negative ones counterclockwise.

Scale Node

Panel: Node Editor Node Composition

Menu: Shift A â†' Convertors â†' Scale



Scale node

This node scales the size of an image. Scaling can be either absolute or relative. If Absolute toggle is on, you can define the size of an image by using real pixel values. In relative mode percents are used.

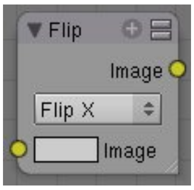
For instance X: 0.50 and Y: 0.50 would produce image which width and height would be half of what they used to be. When expanding an image greatly, you might want to blur it somewhat to remove the square corners that might result. Unless of course you want that effect; in which case, ignore what I just said.

Use this node to match image sizes. Most nodes produce an image that is the same size as the image input into their top image socket. So, if you want to uniformly combine two images of different size, you must scale the second to match the resolution of the first.

Flip Node

Panel: Node Editor Node Composition

Menu: Shift A â†' Convertors â†' Flip



Flip node

This node flips an image at defined axis that can be either X or Y. Also flipping can be done on both X and Y axis' simultaneously.

You can use this node to just flip or use it as a part of mirror setting. Mix half of the image to be mirrored with its flipped version to produce mirrored image.

Displace Node

Panel: Node Editor Node Composition

Menu: Shift A â†' Distort â†' Displace

Ever look down the road on a hot summer day? See how the image is distorted by the hot air? That's because the light is being bent by the air; the air itself is acting like a lens. This fancy little node does the same thing; it moves an input image's pixels based on an input vector mask (the vector mask mimics the effect of the hot air).

This can be useful for a lot of things, like hot air distortion, quick−and−dirty refraction, compositing live footage behind refracting objects like looking through bent glass or glass blocks, and more! Remember what HAL saw in 2001:Space Odyssey; that distorted wide−angle lens? Yup, this node can take a flat image and apply a mask to produce that image.

The amount of displacement in the X and Y directions is determined by

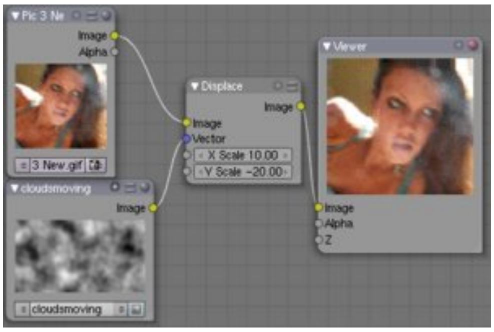
- The value of the mask's channels:
- The scaling of the mask's channels

The (red) channel 1's value determines displacement along the positive or negative X axis. The (green) channel 2's value determines displacement along the positive or negative Y axis.

If both the channels' values are equal (i.e. a greyscale image), the input image will be displaced equally in both X and Y directions, and also according to the X scale and Y scale buttons. These scale button act as multipliers to increase or decrease the strength of the displacement along their respective axes. They need to be set to non−zero values for the node to have any effect.

Because of this, you can use the displace node in two ways, with a greyscale mask (easy to paint, or take from a procedural texture), or with a vector channel or RGB image, such as a normal pass, which will displace the pixels based on the normal direction.

Example



Music Video Distortion Example Using Displace

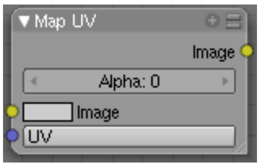
In this example, she's singing about dreams of the future. So, to represent this, we use a moving clouds texture (shot just by rendering the cloud texture on a moving plane) as the displacement map. Now, the colors in a black and white image go from zero (black) to one (white), which, if fed directly without scaling would only shift the pixels one position. So, we scale their effect in the X and Y direction. Upon reviewing it, sometimes stretching in both the X and Y direction made her face look fat, and we all can guess her reaction to looking fat on camera. SO, we scale it only half as much in the X so her face looks longer and thinner. Now, a single image does not do justice to the animation effect as the cloud moves, and this simple noodle does not reflect using blur and overlays to enahnce (and complicate) the effect, but this is the core.

Photos courtesy of Becca, no rights reserved. See also some movies of this node in action, made by the wizard programmer himself, by following this external link

Map UV Node

Panel: Node Editor Node Composition

Menu: Shift A â†’ Distort â†’ Map UV



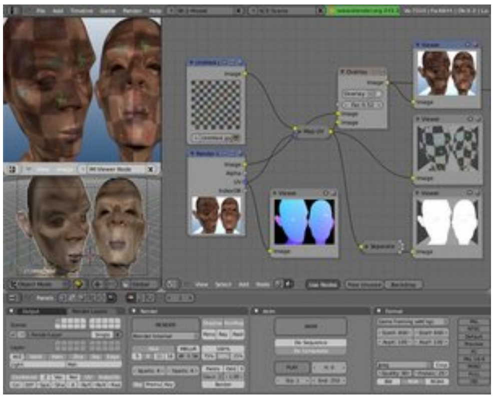
So, I think we all agree that the problem is...we just don't know what we want. The same is true for directors. Despite our best job texturing our models, in post production, inevitably the director changes their mind. "Man, I really wish he looked more ragged. Who did makeup, anyway?" comes the remark. While you can do quite a bit of coloring in post production, there are limits. Well, now this little node comes along and you have the power to **retexture your objects *after* they have been rendered**. Yes, you read that right; it's not a typo and I'm not crazy. At least, not today.

Using this node (and having saved the UV map in a multilayer OpenEXR format image sequence), you can apply new flat image textures to all objects (or individual objects if you used the very cool ID Mask Node to enumerate your objects) in the scene.

Thread the new UV Texture to the Image socket, and the UV Map from the rendered scene to the UV input socket. The resulting image is the input image texture distorted to match the UV coordinates. That image can then be overlay mixed with the original image to paint the texture on top of the original. Adjust alpha and the mix factor to control how much the new texture overlays the old.

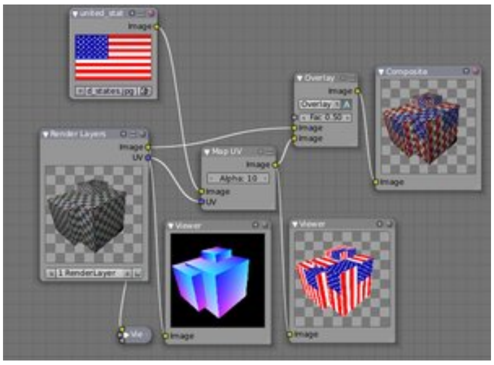
Of course, when painting the new texture, it helps to have the UV maps for the original objects in the scene, so keep those UV texture outlines around even after all shooting is done.

Examples



Adding a Grid UV Textures for Motion Tracking

In the example to the right, we have overlaid a grid pattern on top of the two Emo heads after they have been rendered. During rendering, we enabled the UV layer in the RenderLayer tab (Buttons window, Render Context, RenderLayer tab). Using a mix node, we mix that new UV Texture over the original face. We can use this grid texture to help in any motion tracking that we need to do.



Adding UV Textures in Post−Production

In this example, we overlay a flag on top of a cubie−type thing, and we ensure that we Enable the Alpha premultiply button on the Mix node. The flag is used as additional UV Texture on top of the grid. Other examples include the possibility that we used an unauthorized product box during our initial animation, and we need to substitute in a different product sponsor after rendering.

Of course, this node does NOT give directors the power to rush pre−production rendering under the guise of "we'll fix it later", so maybe you don't want to tell them about this node. Let's keep it to ourselves for now.

Previous: Manual/Compositing Nodes Mattes	Contents	*Next:* Manual/Render Layers

User Manual: Contents | Guidelines | Blender Version **2.45**

As the song goes, birds fly, ships sail...namely, things move; it just comes natural. But maybe you want some things to move *really fast* while other things stay almost the same (think Flash of the Fantastic Four). Especially in anime, the guy always seems to run so fast he's a blur while the girl just stands there and waits for him to run off his excess energy. That, or the good ninja runs up to fiercly chop the bad guy, but the bad guy just stands there.

All great ideas, but...rendering all this motion of all the objects in your scene with motion blur turned on is perfectly feasible...if you have enough CPU and memory. Particularly during the early creative stages, you'll want to see just the stuff that's moving really fast, to see if it looks, well like, really cool.

The Blurry Background

A little backround info, if you will indulge me, to level set your perspective on blur. Three things: eyeball biology, film and CG:

Eyeballs: If you *look into my eyeball*, you will see (pun intended) that my brain processes about 15 images from each eye in parallel each second. My brain cognates those images together and I perceive motion by comparing the two. If something is moving fast enough, I percieve it to be a blur (either because my rods have some latency in reacting to light, or my brain, in overlaying and differencing the images, somehow merges them in a mix sort of fashion). The POINT IS, I *percieve* a motion blur.

Film: to keep us from seeing jumpy motion pictures, we simply doubled the frame rate to 30 frames per second (fps) (24 fps EU). So, the shutter is basically open for a 30th of a second and the film is exposed to the world for that length of time. As things moved in the real world during that time, the film exposure caused the image of the moving thing to be physically blurred or smeared on that frame. When developed and shown, we physically see an image that is blurred. The POINT IS, I *see* a blurred image.

CG: In CG, when a frame is rendered, the computer knows exactly where everything should be, and renders it as such. From frame to frame, an object is location A in frame 1, and location B in frame 2. When we show you these two frames at speed (30 fps), the image appears jumpy to us, because, somewhere between the eyeballs and the film, there isn't that same blurring as the real world and film, and we can tell.

So, how can we make a blurry CG image? Blender has two ways of making a blurry image that is true to the motion of the object: Motion Blur or Vector Blur.

Motion Blur via MBlur button

This is the tradtional way (read: old, outdated) of making motion. Simply enable the MBlur button, located below the Render button (Render button is located on the Buttons window, Scene buttons, Render sub−selection, Render panel). When enabled, Blender uses the motion in the scene to render the full image containing EVERYTHING up to 16 times: for each final frame at time X, he renders frames at X+0.0625, X+0.1250, X+0.1875, X+0.2500, X+0.3125, X+0.3750, X+0.4375, X+0.5000 (for a blur on 8 frames, and a Bf of 0.5), recalculating the exact position of each moving object at each of those exact times, and then finally mixing all those renders together. Great, if you have the time. Most of us don't. So read on.

Overview

Vector−based motion blur for a single image in Blender is accomplished in these broad steps:

1. Render one frame of a scene (or image)
2. Move the object(s) that should be blurred
3. Repeat the previous steps until the range of motion for the blurry objects has been covered
4. Have Blender calculate the motion blur of the objects between each pair of frames
5. Blender combines the individually rendered frames into a final picture

These steps, expounded on in the following subsections, provide a very quick overview of how to get something blurry in as few steps as possible.

Object Motion

There are at least two ways to set up an object's motion:

- Manually change the object's position, from one frame to another, or
- Force the object to follow a path

To manually change an object's position:

1. Add a 'Sphere' and tab out of edit mode.
2. Insert a Location Ipo to position it there at that frame
3. Press the up arrow to advance a few frames (10 by default)
4. Grab the Sphere and move your mouse to drag it somewhere else. LMB click to drop it.
5. Insert a second Location Ipo to position it there at that frame

The sphere will then move from point A to point B during the animation. However, moving the object along a path, using Blender to compute that motion, is conceptually easier than moving it manually between frames. To force an object of follow a path:

1. Add a 'Sphere' and tab out of edit mode.
2. Add a 'Bezier Circle' and tab out of edit mode. The circle will be selected.
3. Make the circle a path by clicking the CurvePath button in the Curve and Surface panel in the Editing Buttons
4. Shift−Select the 'Sphere' via Shift RMB
5. Make the Parent Path of the selected Sphere via Ctrl P to make the sphere a Parent of the circle.
6. In the Object buttons, Constraints panel, add a constraint that the sphere must Follow Path, and enter the name of the circle (usually "CurveCircle") in the OB: field.

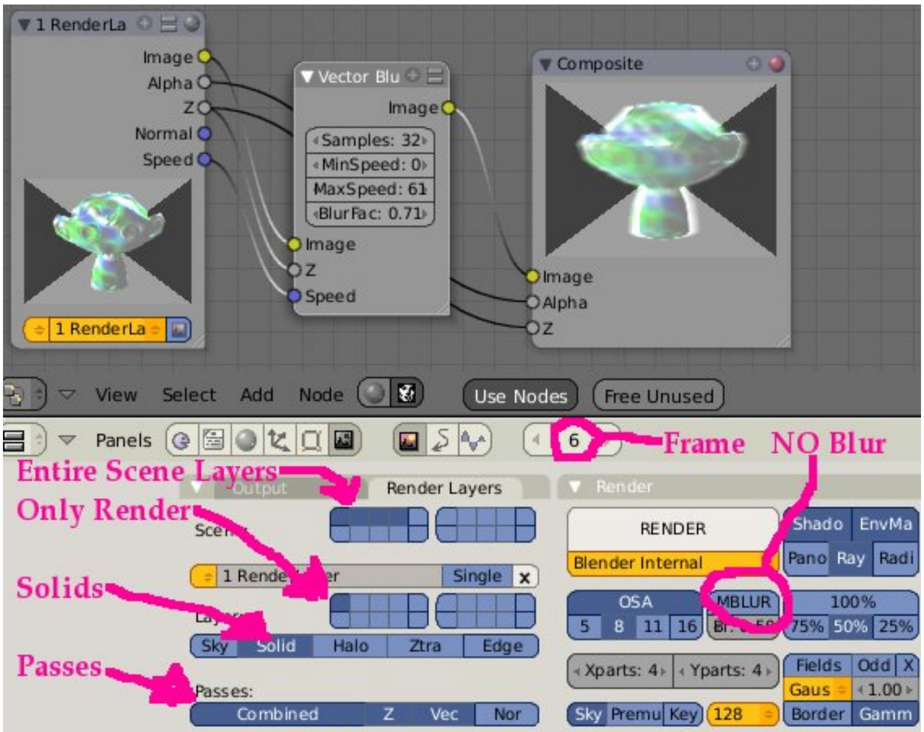
The Sphere should then cruise around the circular path defined by the Bezier Circle if you set your animation to a second or two.

For either method, you need to then:

1. Click the **Scene** button
2. Set the start and end frame of your animation in the Scene Anim panel
3. Click the **Render Layers** tab
4. Enable Passes: **Vec** for Vector Blur use.
5. Do NOT enable **MBLUR**, as we will be using Vector Blur instead.

Configure Blender for Blurring

Blender must be told to blur the objects in motion. This can be accomplished as follows and as shown in the noodle picture below:



Mini blur map

1. Change any window to the **Node Editor**
2. Enable **Use Nodes** and select the Compositing (face) button.
3. Add a **Vector Blur** node (**Add >> Filter >> Vector Blur**)
4. Thread the **Vector Blur** between **Composite** and **Render Layer**
5. Connect **Image** to **Image**, **Z** to **Z** and **Speed** to **Speed** (from **Render Layer** to **Vector Blur**)
6. Connect **Image** to **Image** (from **Vector Blur** to **Composite**)

Render "Animation"

At this point, you should be able to select **Render >> Render Animation** (or **CTRL−F12**) to tell Blender to begin rendering the individual images. Each image will appear slightly different than the previous (sometimes, if the moving object has not been told to move far or fast enough, it will hardly appear to have moved at all!). Once Blender has finished rendering each of the images for the scene, it will merge them together to create the objects in a blurred motion.

Using Vector Blur

Voila, the master Blender coders have given you the tools to realize your creative dreams, whether you're doing anime, mixing real−world footage, or simply making a really smooth animation, and simply can't afford a Cray computer with its 20,000 AMD dual−core Opterons. At least not on this week's allowance. Enter the Vector Blur node: the ability to render motion blur on only a small subset of objects in your scene.

In the Node Editor window above, you see three nodes: The RenderLayer input feeding a Vector Blur Node which in turn feeds a Composite Output Node. What isn't shown is that in frame 2 Suzanne is on the left side, and at frame 11 she is on the right side. The frame being composited is frame 6, as shown in the frame indicator. So, in frame 6, she need to be blurry as she streaks across the screen.

The entire scene contains Layers 1−5 (as shown with those 5 layer buttons enabled), but the RenderLayer is only rendering objects on Layer 1 as shown by the layer set of buttons below the RenderLayer name. As shown in the input node, this is only Suzanne, and she is looking pretty. The Image, Z, and Speed sockets are connected from the Input to the Vector Blur node. The resultant computed output image is fed to the Composite output. The Alpha and Z channels from the Input are threaded to the Output sockets (they go under the blur node, but don't let that throw you).

Down in the Buttons window, on the RenderLayers tab, you need to select at least the Solids option, and the Z and Vec(tor) passes. Enable Combined to speed things up a bit.

The composite output viewer node shows you what a blurry Suzanne looks like, and will look like in the final rendering. Note that the old Motion Blur button is turned off. If we had other RenderLayers set up fo the other Layers of objects in our scene, we could mix those images with the blurry Suzanne to composite our final image.

Other Usage Examples

In the ninja anime example above, using traditional anime technique, you want to freeze the rig of the good ninja and blur the background. To do the more traditional style, put the background in its own layer and create a renderlayer just for it. Make it really move by adding an IPO that covers a few frames. Choose the middle frame as above, route the Renderlayer input through a Vector Blur Filter node, as shown above. Render another image of the static ninja rig, everthing else without motion blur, mix them together over time (like a second or so − 30 frames), and you have your finished product in a minimum of CPU usage. And you have some of your allowance left over for a new pair of sneakers.

Previous: Manual/Render Layers	Contents	*Next:* Manual/Depth Of Field

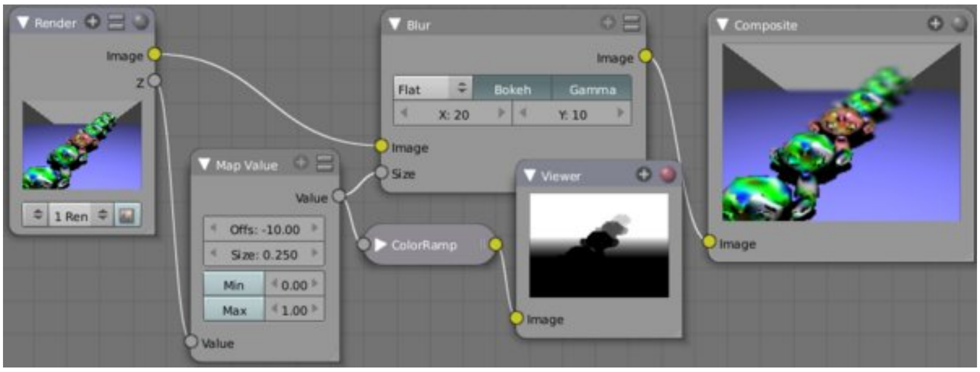
User Manual: Contents | Guidelines | Blender Version **2.42**

We can easily use Blender's compositing nodes to blur the background of our scene. The concept is to take information about our scene, specifically how far away objects are from the camera, and route that information to a blur node that only blurs objects that are beyond a certain distance from the camera.

You will hopefully recall that the RenderLayer node, as well as an Image node with an OpenEXR format picture loaded, puts out a Z depth map. This map specifies how far each image pixel is from the camera. The vector Map Value node is the key here. Ultimately we want to route our Z information into map node that produces a scale from 0.00 to 1.00; the more an object is in the background, the higher its value. If you were to look at this as a picture, you would see that an object in the background gets grayer (the farther back it is). We combine that with a blur factor to blur our original image.

Blurring the Background of a Scene

To blur the background, simply set up the compositing node map shown below:



Background Blur Node Map

This node map shows you the degree of blurriness as a black and white (BW) image in the viewer, and the composite effect on the image in the composite output node. The RenderLayer, which is showing all six monkeys, puts out the image to a blur node. The Z map is routed to a Map Value node, which feeds the Size socket of the Blur and a ColorRamp. The ColorRamp enables you to see the z values in the viewer, and can be deleted along with the viewer.

In the Map Value node, use the Offset to specify that all objects closer than x (in this case, 10) blender units from the camera will get black and thus will not be blurred (recall that black is zero, and zero times any blur is zero). Everything else is therefore the background. Change the offset to move the focal plane. From there (10 units away) back to infinity, the Size value says how fast the background objects will fade away and thus be blurred.

Controlling the Amount of Blur

To calculate Size, subtract the Z−value of the object in your focal plane from the Z−value of the farthest object. This result is the depth of your blurring. Invert this number (divide 1 by X, or 1/x) to yield the Size. The farthest object away will get the full blur effect. Objects in between will get some percentage blur, depending on how far back they go.

The Blur node X & Y offsets say in effect what your f−stop is; namely, how much the most blurry background object is. The sampling filter says how fast they fall off, and which one you choose is a personal preference based on the dramatic impact you want to achieve, much like choosing between lens manufacturers. The Cubic sampling filter may be the most realistic, the Mitch sampling filter gives a nice smooth falloff while preserving crisp highlights, and the flat is dramatic and thus suitable for commercials.

Shutter Speed Effect

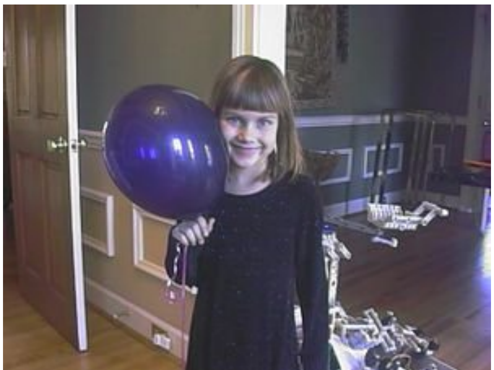
While X and Y are normally set equal to each other, you can achieve a **shutter speed effect** by setting Y to half of X (for capturing horizontal motion like a runner) or the inverse for a jumper (vertical motion). This effect mimics what happens when a photographer tracks the runner (in focus) as they run. The background is not only out of focus, but smeared as the shutter was open while the camera was moved laterally. The example above has the normal blur set to 10, but X is doubled to achieve the shutter speed effect. Obviously, to mimic a faster runner or slower shutter speed, increase the proportion of X to Y.

Good enough?

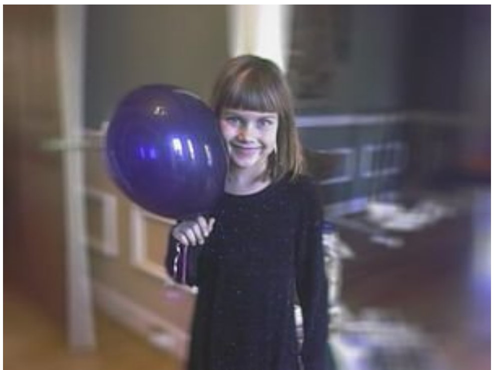
Most animations and stage productions have the actors up front against a background. So, the node setup that you have so far is good enough for most animations.

Blurring a Photograph's Background

When working with still images (e.g. pictures in JPEG or PNG formats), you can still blur the background using Blender. The original photo taken (below, left) shows all objects in focus, even the distracting ones in the background. Using Blender to create a mask allows you to achive the special effect shown next to it.

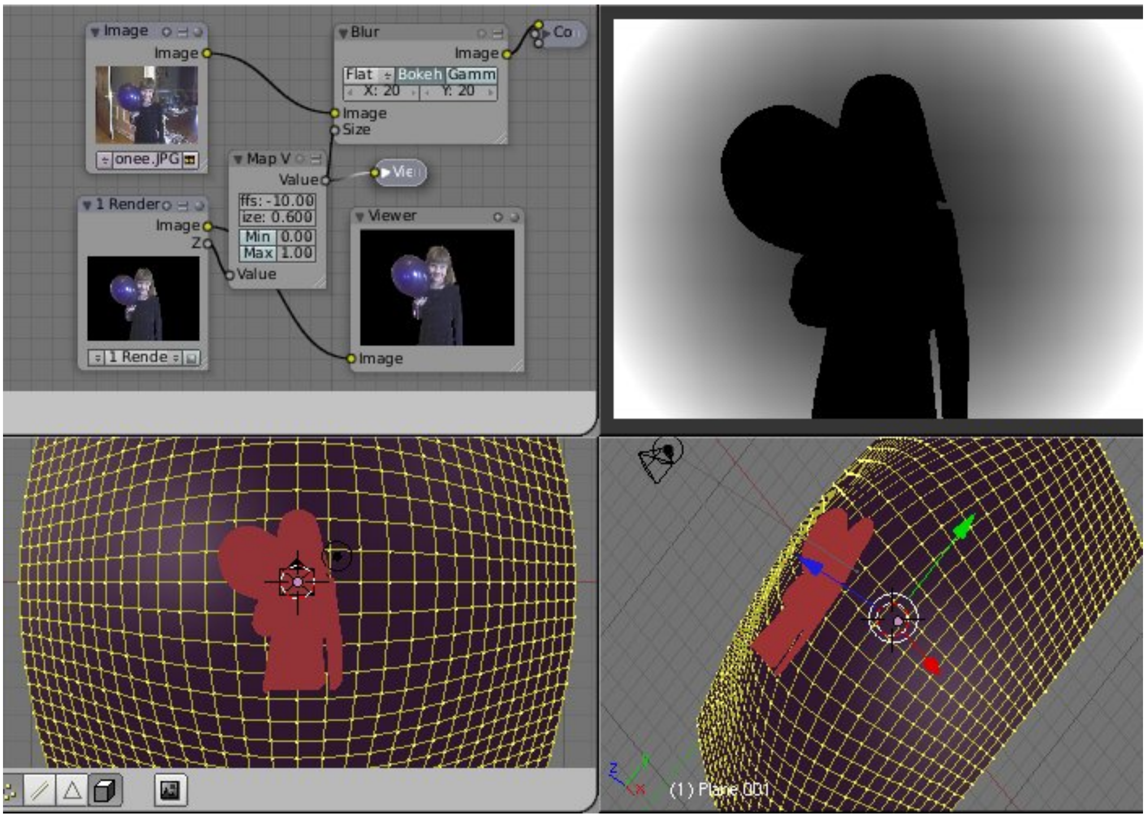


Original



Using a Blur Mask

Use Blender's mesh modeling tools to construct a mask and a backdrop. In the example below, the mask was constructed using a cyclic bezier curve, and the backdrop was a subdivided plane bulged by using the "To Sphere" manipulator at 50%. In this way, the Blender Scene is used to supply a set of Z depth values which, when used as the Blur factor on the photograph, yields exceptional results. A partial screen capture of Blender in action is shown below. In this picture, you can see a node map very similar to the previous one, except that the RenderLayer input supplies the Z values and an Image input node supplies the picture. The black and white image in the UV/Image editor window shows you the Z values as shades of gray. The mask is flat and exactly 10 units from the camera; hence the Offset value. The bulge is just behind it and curves backward and away from the camera, giving a nice smooth falloff.



Blender windows to Construct and Use a Mask and Backdrop

Traditionally, the edges are problematic since the mask invariably has some background pixels. Using a flat backdrop provides uniform blurring of the background, but some of the blurring bleeds into the edges of the mask, resulting in a very blurry edge. The bulged plane allows some blending of the pixels at the edge of the mask, and more as the background expands outward. The mask's edge pixels are smoothed out and blurred into the background to make them unnoticeable. The increasing blurriness as you move out of center of frame is also more realistic and heightens the focus and impact of the crisp center.

Previous: Manual/Vector Blur	Contents	*Next:* Manual/Depth Of Field

User Manual: Contents | Guidelines | Blender Version **2.42**

Depth Of Field (DOF) Explained

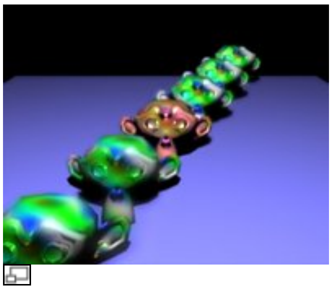
Real world camera lenses and your eyeball transmit light through a lens (cornea) that bends the light, and an iris that limits the amount of light, to focus the image onto the film or retina. Because of the interaction of the lens and iris, objects that are a certain distance away are in focus; objects in the foreground and background are out of focus. We call this distance their depth, or "Z" distance from the camera or eye. Light comes to the lens (in the real world) at an angle; from some direction. What you see depends on your perspective; if you move closer, different angles of the scene are revealed. To make "flat" pictures, like an architectural drawing or plot, Blender can also make an orthographic rendering. So, there are two kinds of renderings, Perspective and Orthographic. Perspective simulates light coming in at an angle to the lens from the field of view, and Orthographic (disabled by default) simulates light coming straight in to an infinitely large backplane or flat retina.

Depending on the diameter of the iris, there is a range (of distance) where objects are in focus. In cameras, the diameter of the iris is controlled by an "f−stop". Said another way, there is *field* of view that you see left to right, up and down; your "picture", if you will. At a certain range, or *depth* away from your eye, things are in focus. For example, at night, you may be able to focus your eye on objects that are 10 to 15 feet away. Anything closer than 10 or farther away than 15 is blurry. Your **depth of field** is thus 5 feet. The larger the iris, the smaller the depth of field. This is why, during the day, you can focus on a range of things stretching out far from you. In film, there is a person whose job is to measure the distance from the camera to the actor's nose, to ensure that the focus is set perfectly. The more that an object is out of its depth (the perfect value for this depth is called *focal plane*), the blurrier it is. In fact, the depth of field is the range on both sides of the focal plane in which the blurriness of the objects is considered to be low enough to be imperceptible. In Blender, this distance is called the Dof Dist or Depth of Field Distance and is set in the Editing buttons for the camera. Alternatively, you can have the camera automatically stay focused on an object, by entering the name of the object in the Dof Ob:: field.

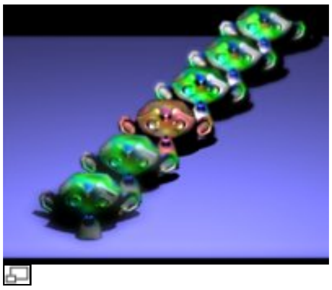
Field of View and Lens Size

The field of view varies by the size of the lens. With cameras, a 35mm lens is kind of a standard size because the picture it takes mimics the size of the picture seen by the eye and pictures can be taken rather close. In Blender, use the Camera Settings to change the size of the lens (35mm is the default). A longer lens taking a picture farther away has the same field of view, but has a different perspective of the view that many directors love because it 'condenses' the scene and smooths a sweep, since it is farther away from the action:

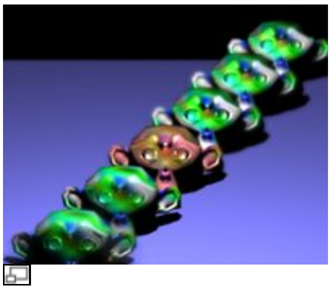
 Concept to Achieving DOF



35mm lens from 10 units away



210mm lens from 60 units away at same location/rotation



210mm at 50 units; repositioned to frame the view similar to the 35mm shot

Zooming in Blender

Zoom is the ability to expand a subset of the picture; we humans have no such ability. Well, I take that back; we do: we just get up off the couch and walk up closer to what we want to see. Blender allow you both actions: you can move the camera closer to or farther away from an object for a track (or "truck") in/out, and/or change its lens size. You can automate these by assigning an Interpolated (IPO) curve to the object or to the camera, respectively.

Depth of Field in Computer Graphics

In computer graphics (CG), there is no physical lens or iris, so the depth−of−field (DOF) is infinite and all objects are always in focus. However, for artistic reasons, we want our main characters to be in focus, and everything else a little blurry, so that our audience does not focus on distracting things in the background. Also, it is easier to discern the main actors when they are in focus, and everything else isn't. So, we have to create an effect, or **Depth of Field Effect**, to composite our images and post−process them to achieve realistic−looking results.

Concept to Achieving DOF

The concept is to take information about our scene, specifically the Z values, and use it to blur objects that are out of depth both behind our DOF, and in front of our DOF. The more out of depth they are, the more they are blurred. We then *combine* those two pictures.

Tools in Blender

Blender 2.43 will have a Defocus node, which will do away with the need for the the noodle described in this section. However, the information on selective blurring may prove interesting. See the linked section for examples on using the Defocus node.

Old Skool (Version 2.42)

Ultimately we want to route our Z information into a foreground Map Value node and a background Map Value node; the result of each node's output being a gray scale (if you looked at it) that ranges from black (0.00) to white (1.00) the more an object is out of depth. We thread that to a Blur factor to blur our original image.

 Combine with a Blurred Background

You will hopefully recall that the input RenderLayer node puts out a Z depth map. This map specifies how far each image pixel is from the camera.

The vector Map Value node is a key node to transform that Z depth map into something the Blur node can use to gradually blur things the more they get out of depth.

The color Z−Combine node combines two images based on which is in front of the other, using the Z−values supplied by two renderlayer nodes.

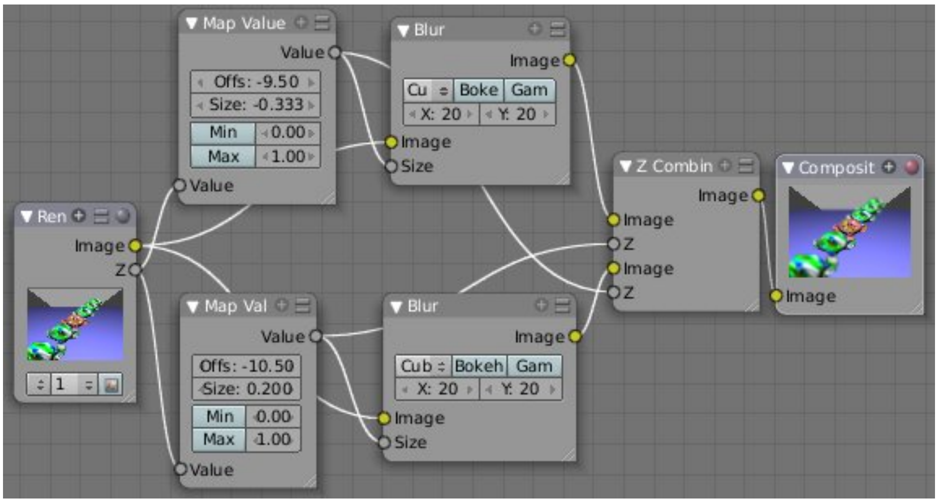
Blur the Foreground

You may recall the previous topic told you how to blur the background of your image. In that topic, we saw that, to start blurring objects a certain distance away from the camera, for example 10 units, we used the Offset to subtract 10, giving a blur factor of zero (no blur). We used the size value as a multiplier to scale the Z−depth values from that zero out to 1.00 (maximum blur).

We now want to blur objects that are closer to the camera, starting with objects, for example, that are 10 units away. The Offset subtracts a value from the Z−depth, and the Size factor multiplies it by some value. So, if we subtract 10 from an object that is 5 away, we get −5.0. Therefore, use a negative Size value to turn that negative into a positive. When we route those values to a Blur node, it will blur objects in the foreground.

Combine with a Blurred Background

Border Select and Shift Duplicate the previous topic's node map (Blur the Background), and plug it in as shown:



Blurring Foreground and Background for Full DOF Effect

Here, the original noodle (shown on the bottom of the image) blurred the background. In the foreground−blurring noodle at the top of the image, the Size has been changed to a negative, and the two blurred images are mixed via Z−Combine. The background, from Purple Suzanne's nose to the sixth monkey's ears, is 5 units; 1/5 = 0.200, the Size for the background blur. From Suzanne's nose to the front monkey's head is 3 units; 1/3 = 0.333, the Size value for the foreground blur.

Why Does It Work?

Notice that the mapped values from the background feed the foreground Z, and the mapped values from the background feed the foreground Z. We do this because in the blur−foreground noodle, the Map value node calculated *negative* numbers for objects in the background. The blur−background noodle calculates *positive* numbers for objects in the background. So, while the positive values tell the blur node what to keep and blur, the negative values tell the Z−Combine node what pixels to use from the image; namely, use the blur−background pixels when compositing the background (since a negative number is *less than* a positive number). Z−Combine thus uses the blurry foreground part of one picture, and the blurry background of the other.

Depth of Field

In the example picture above, the depth of field is −9.50 minus −10.50, or 1 unit deep. Yes, My Dear Aunt Sally, a negative minus a bigger negative is a positive.

Working with the Map

You can vary the size, blur factors, and mixing methods of the foreground and background independently to enhance the impact of the image. Changing the offset changes the focal plane. Spreading apart the offsets between the foreground and background Map Value nodes increases the depth of field. Using a larger Size value increases the rate at which objects blur, and increasing the X & Y values of the blur simulates the f−stop on a real camera.

If you move the camera and/or objects in the scene, you will have to calculate new offset and size values.

Lighting plays an important part

Keep in mind that lighting also plays an important part, and that a spotlight should be trained on the actors in focus. In the real world, adding light to a scene allowed the cameraman to stop down the lens, resulting in a larger DOF without overexposure. You can simulate this just by increasing the difference between the offsets.

Music videos in particular lag the stopping down from the increased lighting, resulting in a sort of 'fade−in−from−white while increasing DOF effect' which is very catchy. You can simulate this just by animating your lamps to reduce energy while increasing your DOF using the offsets.

Adding on to the Effect

It is also possible to add on other nodes to sharpen, enhance, highlight, and/or colorize the foreground or background. Different settings and node maps modifications will have better results depending on the scene setup, the shape of the objects being blurred, and what you want to show in focus. The output Viewer node can show you mapped values as an image; thread the Map Value output socket to the Image socket on a Viewer node, and you will see a gray−scale representation of the mapped values, with black being zero or less, white being 1.00 or more, and shades of gray in between.

Keeping your Desktop Uncluttered

To save window and desktop space, remember that all nodes can be collapsed and moved closer together. The window may be zoomed and panned as well. The UV Image Editor window can show the Viewer node output a little larger for your inspection by selecting Viewer Node as you browse IM: choices.

Excluding Objects from the DOF Effect

To exclude some objects in your scene from the DOF compositing action, you must move them to a different RenderLayer, and then mix the results of this node map with the other RenderLayer input. You can use this really wild effect to have a foreground actor in focus in the middle of a blurred crowd, with a single (ominous) actor far in the background but still clearly in focus. Of course, you can have multiple focal planes within the same image by simply dividing your scene up into renderlayers and applying the DOF effect to each RenderLayer.

Getting Exact Focal Plane Measurements

There is a Caliper script that measures absolute distances between two objects. Use this script to measure the distance from the camera to the object you want in focus. Use this distance as the average of your Map Value offsets between foreground and background.

Applying DOF to Animations

The noodle presented above takes an un−blurred Z−buffered input and blurs it. You can then F3 save your image. However, you may want to perform the DOF compositing later. If so, render your image frames in a format that captures the Z−buffer information but do **not** Do Composite; instead simply save your individual images. Well, you could Do Composite but use other composite nodes to do wonderful things.

Blender outputs the result of a render according to the format specified in the Format panel of the Render buttons. To apply DOF later, you simply must save your images in a format that supports a Z−buffer:

- Open EXR (be sure to click Zbuf and RGBA)

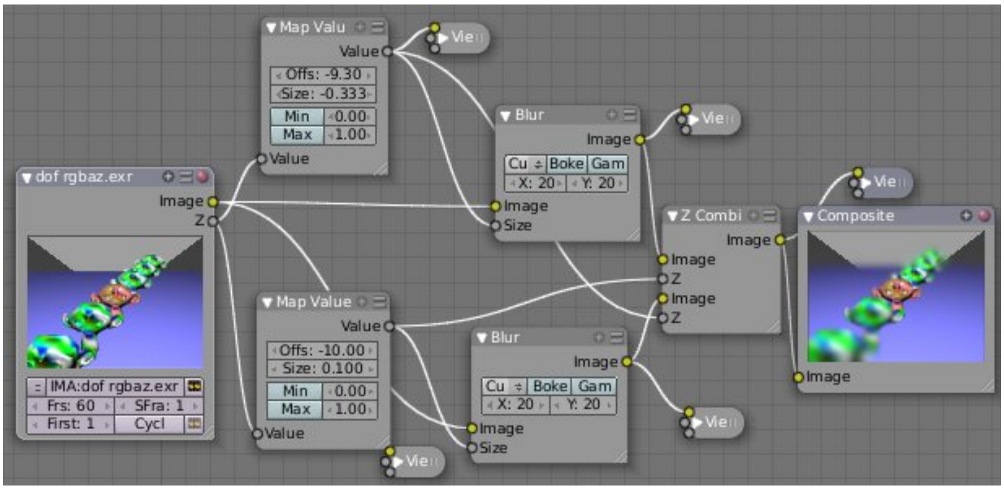
Save EXR space

You may click Half (16−bit) format to save disk space.

You **cannot** use any motion picture codec, because (as of this writing) no motion codecs (AVI or Quicktime) capture alpha or z−buffer information. 'Flat' image formats (JPG, PNG, GIF, BMP, Targa, TIFF) also don't capture Z. While some image formats purport to support a Z−buffer they don't work:

- Radiance HDR
- Iris+Zbuffer

When you are ready to DOF, use the Image input node and specify your image sequece. Set up the DOF noodle shown below, Do Composite, change your render save location, and click the huge ANIM button. Presto, like magic, Blender will zip through your images, applying the DOF in one fell swoop as shown below.



Applying DOF to an EXR image sequence

Two Passes

You CAN use the HDR format by saving two images for every frame: one color (RGBA) and another Z. To do so, you must use a composite noodle that threads the renderlayer Z output to the Composite image socket. Render one pass of 'white' images which contain the Zbuffer information. Render another pass with renderlayer image to Composite image. Then, when ready to DOF, input Image set Z as the Z, connecting the Image output socket to the Map Value input socket. Use input Image set RGBA as the image source to the Blur image input socket.

Previous: Manual/Compositing Blur the Background	Contents	Next: Manual/Compositing Photo Enhancement

User Manual: Contents | Guidelines | Blender Version **2.42**

Photo Enhancement is a term that generally means performing a series of operations on an image to make it appear better−looking. As previously mentioned, during post processing, the director may wish to take a shot that was filmed during the day, and make it appear as though it was nighttime.

This section covers a few of the common uses for nodes in processing images. I wanted to call this page "Making bad photos taken by crappy digital cameras under poor lighting conditions look Good", but the editors wouldn't let me, but you get the idea.

Color Correction

Bad photo

Depending on lighting and materials, you may find that after hours of rendering, the image is composed perfectly, but the colors just don't look right. Rather than make some adjustments and re−render, you can use compositing nodes to quickly correct an image or a series of images. Blender is blindingly fast at performing these operations on images; much faster than re−rendering.

Consider the amazingly poor quality picture of a beautiful wife and two charming children to the right. You may click on the expand icon to see it in its full dismal glory. This is an actual unretouched photo taken by the author at night using a crappy digital camera. Some of the things you notice right away are:

- oversaturated reds and blues
- blurry
- lots of dead space on the left side
- nighttime shot probably has other issues with color balance.

Using the compositing node editor, you can add an Image input node to bring in this single image for post−processing. Immediately, the Saturation needs to be turned down, so add a Hue Saturation color node and reduce the saturation. Connecting that output to a viewer shows good results with a Saturation of around 0.90. Reducing saturation too much results in a black and white image, or an image where the colors look washed out. In that case, use a saturation higher than 1.00 to enhance colors.

Next up is focus; the image is blurry as the photographer had a hard time focusing properly because it was night and he was in a hurry to use a crappy camera. Connect a Sharpen filter node to an RGB Curve node so that we can tweak the individual color channels. Reducing Red and Blue, and actually decreasing overal contrast using the Composite channel on the RGB Curve node brings out flesh tones, and reveals the soft green color of the grass. The Compositing node map is shown below:

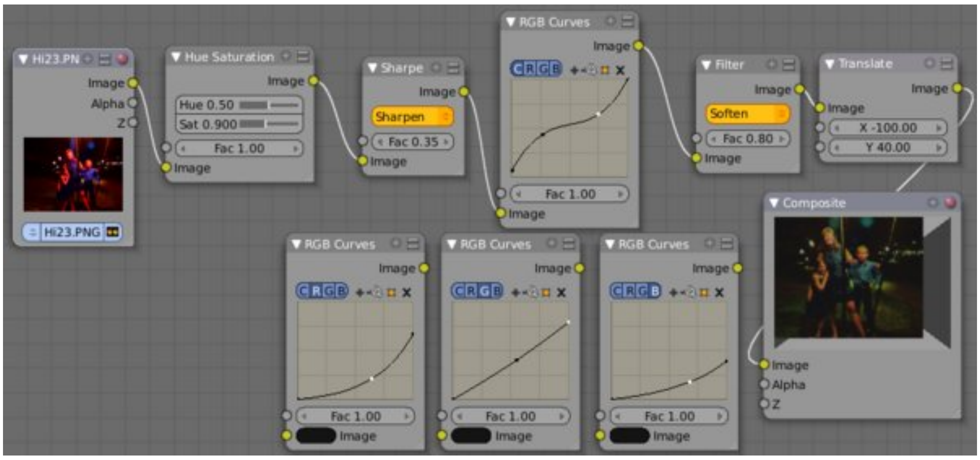


Photo Enhancement Rig

In the node map, the RGB Curves node was replicated to show you each of the Red, Green, and Blue curves. Use the shape of the Composite curve shown to reduce contrast. Usually, when you reduce color in all channels, you have reduced brightness, and you should reduce contrast accordingly.

Saving this .blend file means that you can re−use this rig for every other bad photo you took at night with your crappy little camera. Simply plug in that image, and Blender will instantly perform all these operations. Using Blender in this manner saves you quite a bit of time over using other non−open−source photo editing software.

Better photo

Finally, use the soften filter to blend it back together, and the translate node to perform rudimetary cropping. The final result is shown; a significant improvement!

If this was a movie clip, and the clip was shot under similar lighting conditions, simply using the Image input node controls to bring in the image sequence automatically. Blender will perform these operations on every image.

Previous: Manual/Depth Of Field	Contents	*Next:* Manual/Video Sequence Editing

User Manual: Contents | Guidelines | Blender Version **2.43**

Introduction

In addition to modelling and animation, Blender has a fully functional Video Sequence Editor as well as an advanced node−based editor that also manipulates a video stream. Compositing Nodes operate equally well on images or video streams, and can apply detailed image manipulation on the stream.

Operating at a higher conceptual level, and used later in the video production process, Blender's legacy Video Sequence Editor (VSE) operates on a set of entire strips at a time, as a chunk of footage. The many parts of Blender work together in typical work flow fashion:

1. Model to construct the objects
2. Assign Materials and introduce Lighting to color the objects
3. Animate your objects to make them move
4. Render layers of video using cameras
5. Use Compositing nodes to:
    1. Enhance the images by adjusting colors, adding in−scene special effects
    2. Layer the images into a composite image sequence (strip)
6. Assemble the video strips together to make a movie using the VSE.

The VSE within Blender is a complete video editing system that allows you to combine multiple video channels and add effects to them. Its functionality has been inside Blender since the beginning. Even though it has a limited number of operations, you can use these to create powerful video edits (especially when you combine it with the animation power of Blender!) Furthermore, it is extensible via a plugin system to perform an unlimited number of image manipulations.

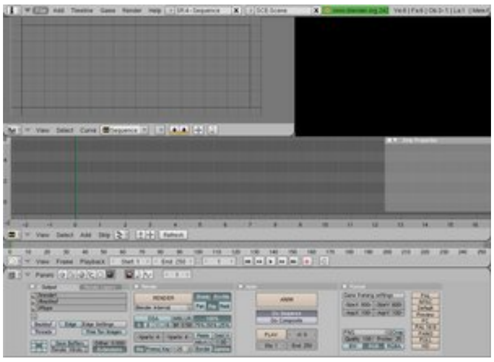
Using the VSE, you load multiple video clips and lay them end−to−end (or in some cases, overlay them), inserting fades and transitions to link the end of one clip to the beginning of another. Finally, add an audio track so you can synchronize the timing of the video sequence to match it. The result of using the VSE is your finished movie.

Previous: Manual/Compositing Photo Enhancement	Contents	*Next:* Manual/Sequence Screen Layout

FFMPEG Support

Support for exporting an avi/quicktime movie using FFMPEG does only work within the Linux and Windows (since 2.44) builds right now.). With FFMPEG support, you are able to save the audio track with your video.

User Manual: Contents | Guidelines | Blender Version **2.43**



Default Sequence screen layout

Blender comes with a few screen layouts, one of which is 5−Sequence. This screen layout is shown to the right. The three main windows are an Ipo window and VSE window on the top row, and another VSE window in the middle, and a Timeline window below that, with the base Buttons windows at the bottom.

The Ipo Window is in Sequence mode; when you animate a sequence, it's Ipo curves will show up here, so you can make fine adjustments to its motion. The black VSE window is set to show you a preview of the finished sequence. Upon opening, it is black because there isn't anything to show.

The middle VSE window, in Sequence mode, is your work area. The VSE window is in the middle, because it is where you will be doing most of your work; being in the middle means your mouse has to travel the least amount of distance, and you are efficient. Keep this in mind when you are arranging *any* screen layout.

The Timeline window below it lets you select where (time−wise) in your animation you want to work, and allow you to quickly set the range of animation you want to scrub. Use the Timeline window's VCR buttons to playback the animation.

Scrub

Just like scrubbing a dirty pot with a brush, where you go back and forth, back and forth, until it is clean, the term *scrubbing* in video is going back and forth over a small range of frames, examining and correcting errors, until it is done.

Previous: Manual/Video_Sequence_Editing	Contents	*Next:* Manual/VSE Modes

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.43**

In addition to being a work area for arranging video strips, the VSE workspace can show you different aspects of the composite result, for the current frame:

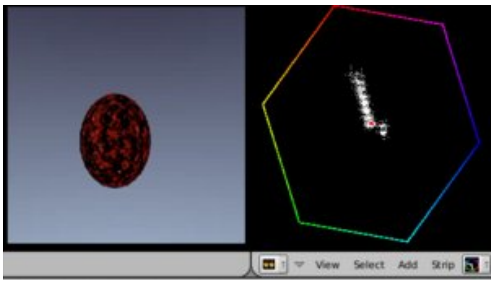
- Chroma: Color hue and saturation
- Luma: Brightness/contrast
- Image: Colors (what you see)
- Sequence: Video strips

VSE Workspace/Image Preview Modes

In the Chroma, Luma, and Image modes, a channel selector appears; channel 0 is the result of compositing the strips with their special effects strips. Channel 1 is what the current frame's image from the strip in channel 1 looks like (channel 1 is at the bottom of the heap). The display of these modes is either the composite (channel 0) or the frame from the strip (channels 1 through n).

Zoom the view of any of these workspaces by scrolling your middle mouse wheel.

Chroma Vectorscope



Example VSE Chroma Preview

For the selected channel, this display shows the colorspace of the image inside a hexagon. Each point of the hexagon is a primary color: red, magenta, blue, cyan, green, and yellow. Black is at the center, and overall saturation is scaled as dots closer to the outside. The example to the right shows that the image has a lot of red (50% saturation) and small amount of blue, with no green.

Use this display to check for too much color saturation. While over−saturated images look great for op−art and computer displays, they stink when shown on the big screen TV. Use the Alt−Animation key to scrub the video; this display will update with a new/revised map for each frame. Just like watching the Image preview to see what it looks like, watch the Chroma Vectorscope to watch for color use.

Luma Waveform

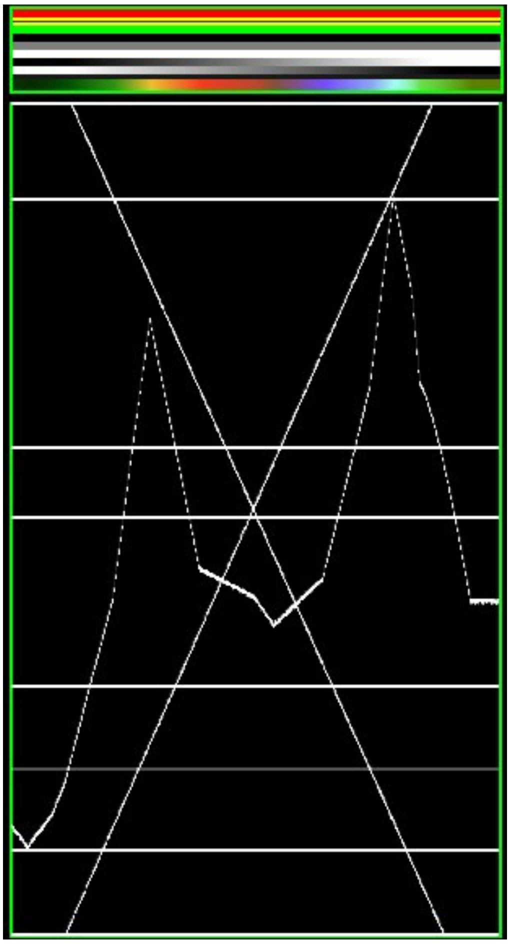
For the selected channel, brightness, or luminosity, is mapped with this display.

Note:

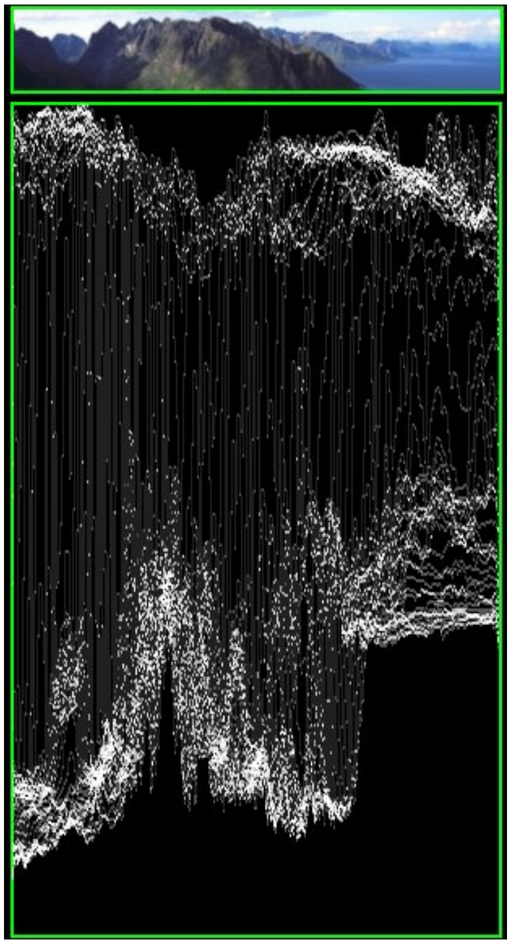
The original explanation seems â€“ to me â€“ quite confused, so I proposes here my own interpretation of this mode (deduced from my experiments)â€¦ If someone knows more about it, please correct it!

In this mode, the vertical axis represents the luminosity: null (full black) at the bottom, full (full white) at the top; the horizontal axis is a mapping from the horizontal axis of the frame. There are as much curves as lines in the frame: each one of this curves represents the luminosity of the pixels of one line. Moreover, the color of a pixel in this mode represents the number of pixels from the matching column of the frame sharing the same luminosity â€“ i.e. the number of curves that cross at this point (black/transparent, for no pixel, white/opaque for at least 3 (?) pixels).

It's not easy to explain, try it with various pictures (rather with natural pictures, the curves are more easy to see) and you might understand better. Look at the two examples below, too.



'Simple' picture.
The various horizontal lines in the Luma waveform match the uniform−color lines of the picture.
Note that the 'grey 20%' one−pixel width line (inside the yellow strip) is represented in the Luma waveform by a grey line.
The two lines drawing an 'X' are from the two linear tone shades (whiteâ†’black and blackâ†’white).



A 'real' picture.
The curves are quite visible.
We found a luma of 80−100% for the sky,
a luma around 40% for the sea,
and a luma of 10−20% for the mountains,
growing around 40% for the sunny part.

Finally, the broken line matches the complex tone shade at the bottom of the picture.

Examples of VSE Luma Previews.
Note that the pictures (first green frame, at the top) are only 50px high, to limit the number of curves displayed in the Luma waveform!

Use this display to check for appropriate contrast and luminosity across all frames in the channel. When spots in the film that should have even illumination don't, it looks like a flashbulb went off or an extra light was suddenly turned on. This can happen if two strips were rendered or shot under different lighting conditions but are supposed to be contiguous.

Image Preview

In the upper right window pane of the Sequence screen layout is another VSE window, this one set to Image Preview mode. It shows you what the resulting video will look like when saved.

Sequence Mode

The main working mode for adding strips and moving them around, cutting, grouping (making meta) and splicing them through special effects.

Previous: Manual/Sequence Screen Layout | Contents | *Next:* Manual/Using VSE

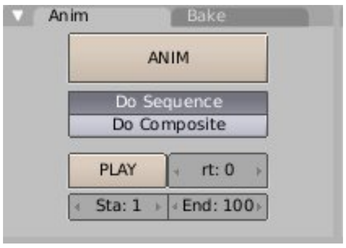
User Manual: Contents | Guidelines | Blender Version **2.43**

Blender's Video Sequence Editor is a flexible workbench for editing your video footage. It is used to review your footage, and glue many sequences of your movie together. It offers a number of built−in and plug−in effects to transition from sequence to sequence, providing advanced hollywood−style effects for a professional looking video.

Overview of the Video Sequence Editor Window

Video Sequence Editor in Sequence display mode

The Video Sequence Editor has a header (where the menu and mode are shown) and a workspace, and works in one of four modes. In Sequence mode, the workspace is horizontally−striped into channels, and each video strip will go in a horizontal channel. Each channel is numbered on the left−hand side, starting from 0 (you can't put anything thing in this special one!) and going up; the picture to the right shows three channels (zero to two). Stripes toward the bottom are more dominant, which we'll get to in a minute. In the x direction, seconds of animation or frames of animation (T to choose) are used as the measure of time (seconds 1 through 7 are shown). You can scale the time using the zoom keys or mouse actions (see the Reference for more info).



Enable Sequence Output

When you click Render or Anim to generate an image or video, Blender has a choice of what image to compose for the current frame/scrub range:

- Current Scene layer result
- Sequence Editor channel 0 result
- Composition Node Editor renderlayer result

Tell Blender to use the output of the Video Sequence Editor by enabling "Do Sequence" in the Buttons Window, Scene Render buttons, Anim panel.

View Menu

As usual, this menu controls what and how you view in the workspace.

Mode: Sequence

Hotkey: Shift space, Ctrl uparrow, Ctrl downarrow

Menu: View −> Maximize Window

Use this when working arranging a lot of strips and you want to use all of your screen to work.

Mode: Sequence

Hotkey: T

Menu: View −> Show Frames, View −> Show Seconds

Time toggles the units of measure across the bottom of the workspace between seconds and frames. Seconds depends on the Frames/second setting in the Scene Render Format panel in the Buttons window.

Mode: Sequence

Menu: View −> Lock Time to Other Windows

Locking Time to other windows means that if you change your position in time (by left−clicking in the workspace to move your vertical green cursor), other windows will update as well to reflect what the video will look like at that time.

Mode: Sequence

Hotkey: NumPad .

Menu: View −> View Selected

Zooms in the display to fit only the selected strips

Mode: Sequence

Hotkey: home

Menu: View −> View All

Zooms (out) the display to show all strips

Mode: Sequence

Hotkey: Shift Alt A

Menu: View −> Play Back Animation in 3D View

Plays the animation for the selected scrub frame range in all windows: In VSE windows, shows you their respective display (Image, Chroma, Luma). In any 3D windows, shows you the objects moving.

Mode: Sequence, Image, Chroma, Luma

Hotkey: Alt A

Menu: View −> Play Back Animation

Plays the animation for the selected scrub frame range in the window.

Select Menu

This menu helps you select strips.

Mode: Sequence

Hotkey: A

Menu: Select −> Select/Deselect All

Selecs all the strips loaded.

Mode: Sequence

Hotkey: B

Menu: Select −> Border Select

Begins the Box mode select process. Click and drag a rectangular lasso around a region of strips in your Sequence workspace. When you release the mouse button, the additional strips will be selected.

Add Menu

Mode: Sequence

Hotkey: space

Menu: Add

This is the main menu you will be using. In general, you load up your strips, create strips of special transition effects, and then animate out your sequence by selecting "Do Sequence" and clicking the Anim button. You can use the Add menu in the header, or hover your mouse cursor over the Sequence workspace and press space.

Clips can be Huge

A three minute quicktime .mov file can be 140Megs. Loading it, even over a high−speed LAN can take some time. Don't assume your computer or Blender has locked up if nothing happens for awhile.

Add Movie and Image (Stills or Sequences)

First, let's add a clip:

- A movie clip in the Audio−Video Interlaced format (*.avi file)
- A movie clip in the Apple QuickTime format (*.mov)
- A single still image to be repeated for a number of frames (*.jpg, *.png, etc.)
- A numbered sequence of images (*−0001.jpg, *−0002.jpg, *−0003.jpg, etc, of any image format)
- One or more images from a directory

- A Scene in your .blend file.

Blender does not care which of these you use; you can freely mix and match any of them. They all become a color−coded strip in the VSE:

- Blue is used for Avi/mov codec stips
- Grey is a single image that is repeated/copied
- Purple is an image sequences or group of images played one after the other
- Green is an Audio track

When you choose to add one of these, the VSE window will switch to a file browser for you to select what you want to add. Supported files have a little rectangle next to their name (blue for images, green for clips) as a visual cue that you can pick them successfully:

- When adding a Movie or Movie+Audio:

  LMB LEFT CLICK to put the name of the file into the text box at the top; this selects a **single** file (like a movie)

- In the case of (numbered) image **sequences**, you have a choice:

  Directory: RMB right−click on a directory name, and all files in that directory will be brought in as part of the image, in sort order, one image per frame
  Range: Navigate into the directory and right−click and drag over the a range of names to highlight multiple files. You can page down and continue right−click−dragging to add more to the selection
  Batch: Shift−right−click selected non−related stills for batch processing; each image will be one frame, in sort order, and can be a mix of file types (jpg, png, exr, etc.)
  All: Press A to select/deselect All files in the directory.

When you click the Select <whatever> button, the window pane will switch back to VSE, and the strip will be rubber−banded to your mouse. You cannot load multiple movies at the same time by right−clicking them; no movies load if you right click them. Right−clicking only works for images.

Error: The selected file is not a movie or FFMPEG support not compiled in!

means that the file is not a movie that Blender can recognize, or **you selected with the wrong button**. You get this error message because you *right*−clicked on a movie file, OR you don't have a codec that can decode the avi file. If it's the latter, find a codec so you can play the file outside of Blender, and then you will be able to load it. If it's the former, you must left−click to select movies.

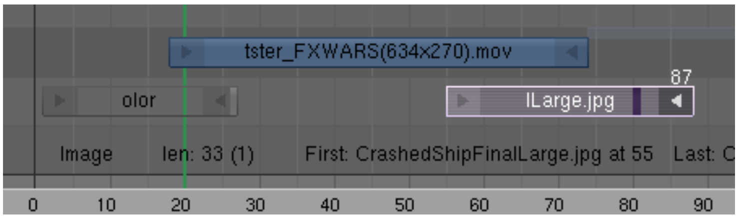
In order to add items to the VSE, left−click for movies, left−click for single images, or right−click and drag for image sequences. Move your mouse to the frame/time and stripe you want, and click to break the rubberband and drop the strip in place (in a channel and starting at a frame).

When you add an image, Blender makes it into a 50−frame strip, which means that image will be in your video for two seconds (at 25 fps â€“ PAL). Aside from re−positioning it, you will want to scale it by RMB −clicking on either the start or end arrow, and dragging left or right. As you move, the frame number updates to say where the arrow is. Click LMB to validate, or RMB to cancel the modification.

Dealing with Different Sizes:

Dealing with different sized images and different sized outputs is tricky. Think like a pixel. If you have a mis−smatch between the size of the input image and the render output size, the VSE does try to auto−scale the image to fit it entirely in the output. This may result in clipping. If you do not want that,

use Crop and/or Translate in the Input panel to move and select a region of the image within the output. When you use Crop or Translate, the auto−scaling will be disabled and you can manually re−scale by adding the Transform effect.



If you scroll up the workspace, you will see an information channel (at vertical location channel 0) that gives you some helpful hints about the active strip. The example below shows a color strip from frames 1 to 25, then a mov file, and then an image strip. The info channel shows handy information about the image strip, whose name has been scrunched in the strip display, but is clearly spelled out in the information strip.

9999 frames go by

Ok, so that was a very obscure reference to a song about 99 balloons, but we really have not anticipated how fast Blender has moved into mainstream video editing. Unfortunately, we initially reserved 4 digits for the filename of each video image sequence set. While that provides for up to 400 seconds of video (about 5 minutes US), with Blender moving into movies, you need to break up video strips into 4 digits only, and others 5 digits (10000−19999), (20000−29999), etc.

Codecs

You must have a codec on your machine that can decode the avi file. Blender does not control these. For example, the XviD codec is available from www.xvid.org

FFMPEG Support

If you are using a Blender build with FFMPEG support, you will be able to load audio and video strips together; select Movie+Audio(HD) and when you drop the strip, the strip will split into an audio and video channel strips.

Add Scene

You can add the virtual image output of a Scene in your current .blend file as well. Select the scene from the popup list, and a strip will be added and rubberbanded to your mouse just like a movie or image. The strip length will be determined based on the animation settings in that scene (not the current scene, unless the VSE is operating in the same scene).

When adding a Scene strip, please note that, in order to show you the srip in the VSE Image preview mode, Blender must render the scene. This may take awhile if the scene is complex, so there may be a delay between the time you select the scene and the time the strip appears. To reduce the delay, simplify the scene rendering by selecting fewer layers to render.

Add Audio

The VSE can incorporate an audio channel which you can hear as you scrub. Add an audio track when you are trying to time your video/animation to an audio track, or vice versa. Please refer to the Audio Sequences

section for more information.

Add Effect

Available Built−in Effects

Blender offers two categories of effects: Built−in and Plug−in. The built−in effects are listed to the right. They are built−in to Blender and everyone has them. The plug−in effects are separate files in a sequence−plugin directory on your PC that are loaded as they are needed. While a standard set of plugins are distributed when you installed Blender, everyone's computer may have a different set.

Mode: Sequence

Hotkey: space

Menu: Add −> Effect

Every Built−in effect is explained later individually, but they all are added and controlled in the same way. To add an effect strip, select one base strip (image, movie, or scene) by RMB clicking on it. For some effects, like the Cross transition effect, you will need to Shift RMB a second overlapping strip (it depends on the effect you want). Then select Add −> Effect and pick the effect you want from the pop−up menu. When you do, the Effect strip will be shown above the source strips. If it is an independent effect, like the color generator (described later), it will be rubberbanded to your mouse; click to drop the strip.

Since most Effects strips depend on one or two source strips, their frame location and duration depends on their source strips. Thus, you may not be able to move it; you have to move the source strips in order to affect the effect strip.

To use an effect that combines or makes a transition between (or composites) two strips, you must Box select or shift−right−click two of them. When you add the effect strip, it will be placed in a channel above the two in Grab mode (click to drop it on a channel). Its duration will be the overlap between the two strips as a maximum.

With some effects, like the AlphaOver, the order in which you select the strips is important. You can also use one effect strip as the input or source strip with another strip, thus layering effects on top of one another.

Note: The only exception is the Color Generator effect. It does not depend on a base strip; you can add and position it independent of any other strip. Change the length as you would any strip.

Mode: Sequence, Effects Strip Selected

Hotkey: C

Menu: Strip −> Change Effect

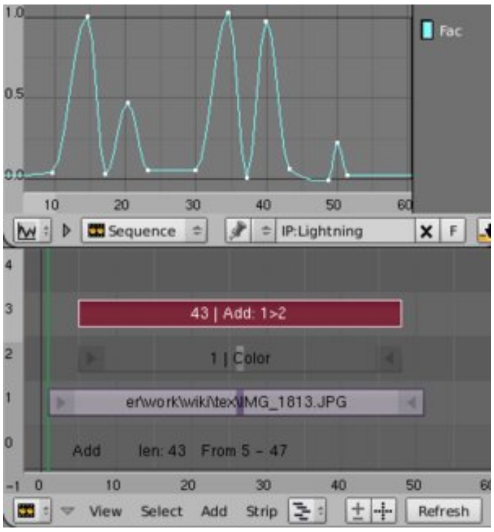
If you picked the wrong effect from the menu, you can always change it by selecting the strip (RMB) and using the Strip−>Change Effect selection. Or, you can press Change to switch effects on a selected Effects strip.

Adding Plugin Effects

Sequence Plugins are special little routines written by special programmers in the C language as a dynamic load library (.DLL). A DLL can be loaded at any time (dynamically) as it is needed, so it "plugs in" to Blender. (In the case you wondered: the extension is platform dependent. These files are named .so (shared object) on Linux e.g.)

The image to the right shows the Sequence DLL's that I have available on my system. Each of them do some special effect indicated by their name or as explained on the Blender Resource Page for Plugins or programmer website. For example, the Iris plugin transitions between two strips by opening an expanding hole in the middle of the first and letting the second one show through, like an iris of a camera opening up. Some of these plugins can be five or more years old and still work very well; Blender tries to ensure backward compatibility, and they should work independent of output format or resolution (size).

Animating Audio/Video Effects



Animating the Add Effect

The degree to which some effects manipulate the image (called a Factor) can be controlled over time. For example, over the course of 100 frames, you can vary the factor from 0 to 1, then down to 0.5, then back up to 1.0. You do this by defining an Ipo Curve for that effect. Some effects (Add, Multiply) can be animated through the Ipo curve; others (Color Generator, Glow) are controlled through the Properties panel.

To the right is an example of animating the Add effect to produce the effect that lightning has. To add control points in the Ipo Window, Ctrl LMB click anywhere, and a Factor curve will be added. In this case, we have added gray to our image in a jagged Ipo set. When (or if you have already) gotten into nodes, the use of this Factor is the same in many nodes, especially the Mix node. By default, when you add a special effect, like a Cross, Blender provides a smooth transition to the new picture, from 0.0 to 1.0, over the duration of the frames (the sfx strip). In the case of an add or multiply or subtract effects, Blender add the effect immediately, and keeps it there; the factor jumps to 1.0 and stays at 1.0. Now, we know how much you all like to control things, so you have the Ipo Window that can operate on a strip, or Sequence. This is why there is an Ipo Window in the upper left−hand corner of the screen layout. In general you:

- Select the effect strip
- Decide whether you want the Ipo curve to be relative to the start of the first base strip, or locked to the global frame number. If you want it locked to the global frame counter, enable the IPO Frame locked button
- Ctrl− LMB click control points in the Ipo window (Sequence mode) to define the transition factor and timing. To make a smooth transition to the final effect, just define one point at the starting frame with a Fac of 0.0, and another point at the ending frame with a Fac of 1.0. You can define a jagged curve with lots of points to get a very dynamic effect.

Not all effects can be animated; The built−in Glow is fixed, but there is an AnimGlow plugin which can be. Add, Multiply, Divide, Subtract, and Transform effects are always animatable.

Animating an audio strip affects the volume of the strip in the resulting composite. Use IPO animation on an audio strip to fade in/out background music or to adjust volume levels. Layered/crossed audio strips are added together; the lower channel does not override and cut out higher channels. This makes Blender an audio mixer. By adding audio tracks and using the IPO curves to adjust each tracks' sound level, you have an automated dynamic multi−track audio mixer!

To make the any effects strip (such as Cross, Wipe, or Transform) that operates on two strips, make its transition over a shorter time period, just shorten its strip by shortening the duration by which the two strips

overlap.

If Ipo Frame locked is enabled, Blender will use the Fac value from frame x in the Ipo window to apply the Effect at frame x in the VSE window. If it is disabled, the Ipo curve is relative to the start of the first base strip. Generally, you want the transition effect to be relative to the strips they operate on, but in certain matching moves or transitions, you can lock it to a specific frame.

Strip Menu

This menu allows you to operate on video strip(s) as a whole.

Mode: Sequence

Hotkey: M, Alt M

Menu: Strip −> Make Meta Strip, Separate Meta Strip

A Meta−Strip is a group of strips. Select all the strips you want to group, and Make them into one meta. The meta spans from the beginning of the first strip to the end of the last one, and condenses all channels into a single strip, just like doing a mixdown in audio software. Separating (ungrouping) them restors them to their relative positions and channels.

Mode: Sequence

Hotkey: x, Shift D

Menu: Strip−> Delete, Duplicate

If you have added a strip by mistake or no longer want it, delete it by pressing X or using this menu option. Duplicate a strip to make an unlinked copy; drag it to a time and channel, and drop it by LMB click.

Mode: Sequence

Hotkey: K

Menu: Strip −> Cut at Curent Frame

While splicing two strips happens just by placing them finish−to−start, cut a strip by pressing K to cut. At the selected frame for the selected strips, K cuts them in two. Use Kut to trim off roll−ups or lead−ins, or roll−downs or extra film shot ("C" was already taken for Change).

Note on the 'cut'

When you 'cut' a strip, you don't really make a cut like it was with the 'old editing' on real film. In fact, you make a copy of the strip: the end of the original one is 'winded' to the cut point, as with the beginning of the new copy.

For example, imagine that you have a strip of **50** frames, and that you want to delete the first ten ones. You have to go to the **11th** frame, and hit K; the cut 'divides' your strip in two parts. You now can select the first small part (frames **1** to **10**), and delete it hitting X.
You might think that you have really erased the frames **1** to **10**, but there are still there, 'winded', as in a film reel, under your frame **11**: you just have deleted one of the tow copies of your strip created by the 'cut'. And you can at any time get your 'lost' frames back (just RMB −click on the left arrow of the strip, then Grab it

to the left to display the desired number of frames again (or to the right to 'hide' more frames â€“ this is another way to remove frames at the beginning/end of a strip!).

This is at the heart of nearly every editor solutions, and that's quite handy!

Mode: Sequence

Hotkey: Shift S

Menu: Strip −> Snap to Curent Frame

Position your cursor (vertical green line) to the time you want. Snap to current frame to start a strip exactly at the beginning of the frame. If your Time display is in seconds, you can get to fractional parts of a second by zooming the display; you can get all the way down to an individual frame.

Mode: Sequence

Hotkey: G

Menu: Strip −> Grab

Moves the selected strip(s) in time or in channels. Move your mouse horizontally (left/right) to change the strip's position in time. Move vertically (up/down) to change channels. Moving a strip above (to a higher channel) and over another strip means that it will not display during the overlap; lower channel strips are displayed "in front" of higher channels â€“ **WARNING: this is exactly the opposite that what happens with most of the other video editors!** To get them both to be seen, you have to mix them using the Add, Multiply, Subtract or other compositing effects.

Mode: Sequence

Hotkey: Y

Menu: Strip −> Separate Images to Strips

Converts the strip into multiple strips, one strip for each frame. Very useful for slide shows and other cases where you want to bring in a set on non−continuous images.

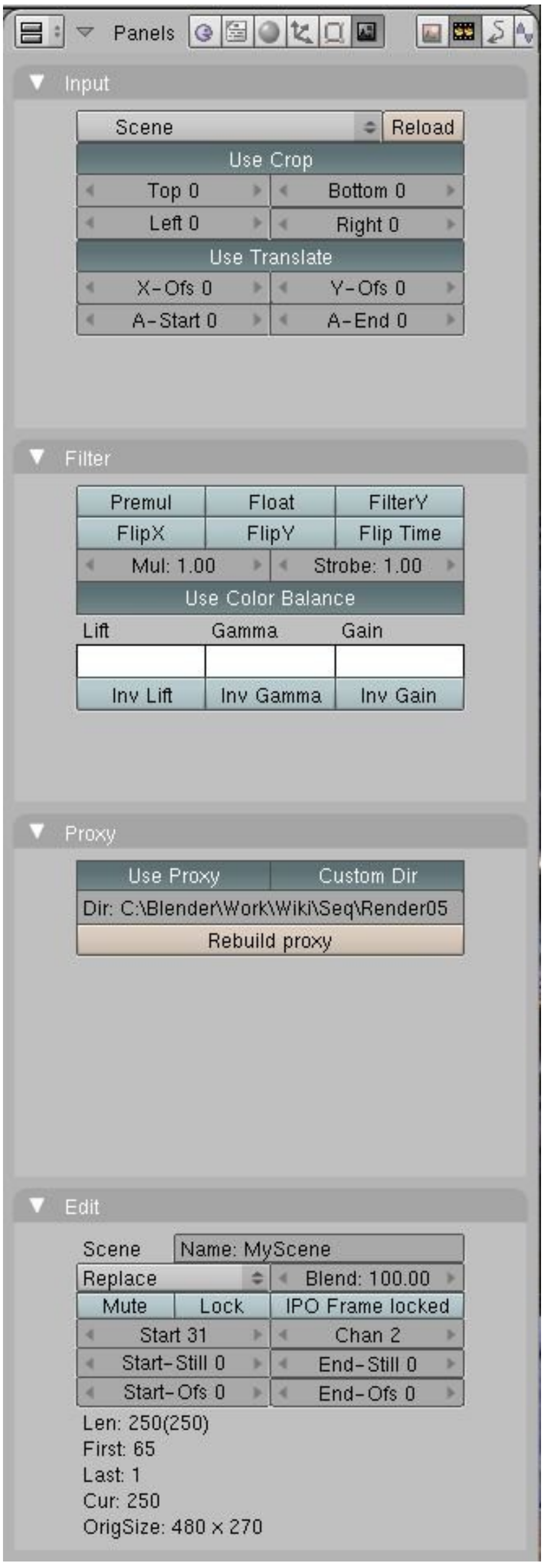
Strip Properties

Blender 2.46 expanded the Strip properties from one panel to an entirely new Render (F10) subcontext.

V.: 2.46

Mode: Sequence, Effects Strip selected

Menu: Buttons Window, Render Context (F10), Sequence sub−context



Strip Properties Panels

The properties for the strip are examined and set in various panels in the Buttons Window, Render Context, Sequence sub−context. Until we get around to documenting all of these new features, please see the [Release Notes.]

- Input − where to pull images from
- Filter − Image pre−processing
- Proxy − Use representatives of the real image, for low−powered PCs

- Edit − change properties of the strip

The panels for each of these sets of options and controls are shown to the right

Input Strip Properties Panel

Controls the source of the strip. Fields include filename, current frame, and auto−crop and auto−translate features, as well as an offset for starting and ending the strip.

Filter Strip Properties Panel

Enables you to quickly set common image pre−processing options.

Premul: Premultiply Alpha
Flip: X − flips (reverses) the image left−to−right, Y reverses top−to−bottom, and Time reverses strip image sequence
Use Color Balance provides three filters to adjust coloration: Lift, Gaussian, and one other. Each pass can have a positive, or inverted effect by clicking the appropriate button. Set the amount of the effect by setting the color swatch; white (RGB 1,1,1) has no effect.

Proxy Strip Properties Panel

A proxy is a smaller image (faster to load) that stands in for the main image. When you Rebuild proxy Blender computes small images (like thumbnails) for the big images and may take some time. After computing them, though, editing functions like scrubbing and scrolling and compositing functions like cross using these proxies is much faster but gives a low−res result. Disable proxies before final rendering.

Edit Strip Properties Panel

Name
: You can name your strips

Blend Mode
: By default, a strip Replaces the output image of any lower−level strips. However, many other blending modes are available based on the strip type. For example, Alpha−Over automatically overlays the image on top of a lower level strip. These autoblending modes obviates the need for separate effect strips. Blend percent controls how much of an (even over time) effect the strip exerts

Mute
: Hides the strip so that it does not participate in the final image computation

IPO Frame Locked
: Uses the Sequence IPO curve to determine the Blend percent.

Start
: changes the starting frame number of the strip, which is the same as grabbing and moving the strip. Tip: when you add a strip, I like to just drop it and then use this field to place it at the frame I want, rather that trying to drag and drop in exactly the right place.

Chan

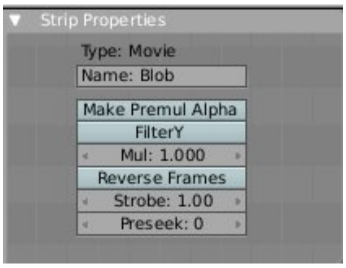
Changes the channel number of the strip, like G−Y.

V.: 2.45

Mode: Sequence

Hotkey: N

Menu: Strip −> Strip Properties



Strip Properties

Press N to display a floating panel that shows you properties for the object selected; in this case a video Strip. By default, the name of a strip shown in the workspace is its filename. You can name strips by LMB clicking in the Name: field and entering a descriptive name; the workspace will display that name.

Use the Convert to Premul button if a strip has an Alpha (transparency) channel. Use FilterY if the strip is from broadcast video and has even or odd interlacing fields. Enhance the color saturation through the Multiply field. Play a strip backwards by enabling Reverse Frames. Tell Blender to display every nth frame by entering a Strobe value. Finally, when using MPEG video (VCD, DVD, XVid, DivX, â€¦), an image is built up over the course of a few frames; use the Preseek field to tell Blender to look backward and compose the image based on the n previous frames (e.g. **15** for Mpeg2 DVD).

For all effects, use the Strip Properties panel to control the effects strip; each effect has different controls, but they can all be set in the Properties panel. Control the length of the strip to vary the speed with which the transform happens. Regardless of whether they are built−in or plug−in, all effect strips do some special image manipulation, usually by operating on another strip or two in a different channel. The effect strip is shown in some channel, but its resultant effect shows up as Channel 0.

Working with Strips

Here are some common tasks that you will want to perform within the VSE.

Scrubbing

To move back and forth through your movie, use the Timeline window. LMB click and drag left/right in the timeline window, moving the vertical bar which indicates the current frame. As you do, the image for that frame is displayed in the VSE window.

Real−time scrubbing and image display is possible on reasonable computers when viewing an image sequence or movie (avi/mov) file. Scene images have to be rendered individually, which may take some time.

Selecting and Changing the Length of a Strip

To edit a movie or image strip:

- RMB in the middle of the strip selects the **entire** strip; holding it down (or pressing Grab) and then moving the mouse drags a strip around.
- RMB on the left arrow of the strip selects the **start** frame offset for that strip; holding it down (or pressing Grab and then moving the mouse left/right changes the start frame within the strip by the number of frames you move it:
    - ♦ If you have a 20−image sequence strip, and drag the left arrow to the right by 10 frames, the strip will start at image 11 (images 1 to 10 will be skipped). Use this to clip off a rollup or useless lead−in.
    - ♦ Dragging the left arrow left will create a lead−in (copies) of the first frame for as many frames as you drag it. Use this when you want some frames for transitions to the this clip.
- RMB on the right arrow of the strip selects the **end** frame of the strip; holding it down (or pressing Grab) and then moving the mouse changes the ending frame within the strip:
    - ♦ Dragging the right arrow to the left shortens the clip; any original images at the tail are ignored. Use this to quickly clip off a rolldown.
    - ♦ Dragging the right arrow right extends the clip. For movies and images sequences, more of the animation is used until exhausted. Extending a clip beyond its end results in Blender making a copy of the last image. Use this for transitions out of this clip.

Multiple selection

You can select several (handles of) strips by Shift RMB −clicking: when you'll hit G, everything that's selected will move with your mouse â€“ this means that, for example, you can at the same time move a strip, shorten two others, and extend a forth one.

- STRIP EXTEND. With a number of Image strips selected, pressing E enters EXTEND mode. All selected strip handles to the "mouse side" of the current frame indicator will transform together, allowing you to essentially extend the strips that fall exactly on the current frame marker and having all others adjust to compensate.
- Additional Selection Methods
    - ♦ ALT RMB and CTRL RMB on a strip will select the Left or Right handle of that strip and the neighboring handle on the next strip. Select with this method to move the boundary between two adjoining strips without affecting the outer limits.
    - ♦ ALTCTRL RMB on a strip will select both handles of the strip, plus the neighboring handles on the immediately adjoining strips. Select with this method to move a strip that is between to others without affecting the selected strip's length.

Action Stops

When extending the start beyond the beginning or end after the ending, keep in mind that only the last image copies, so when viewed, action will stop on that frame. Start your transition (fade, cross) a little early while action is still happening so that the stop action is not that noticeable (unless, of course, you want it to be, like the 80's drama sitcoms).
Change the length of an effect strip by changing the start/end frame of the origin strips.

Zoom, Scale Display and Refresh Header Buttons

Zoom, Scale and Refresh buttons

These buttons are found throughout Blender on a window's header. LMB Click and drag the +/− button left and right to scale the display around the cursor (the vertical green line). Moving left zooms out, and right zooms in across frames. Moving up and down zooms in and out of channels.

Clicking the crosshair puts your mouse in a box−select mode. Select a region of the workspace by LMB clicking and dragging over a rectangular region of the workspace. When you release the mouse, the workspace display will zoom in to fit that region in your entire display.

On lesser powered−PCs, Blender may defer window update until you stop working and it gets some CPU time to work behind the scenes doing computation. In this way, Blender does not make you wait on it; it waits on you and gives you precedence. Click the Refresh button to force Blender to update and synchronize all displays to the current frame.

Reverse Action

In the sequence buttons in the scene buttons (F10), select "flip time" in the "filters" panel.

Slow Motion, Fast Forward, Time Warp

Use the built−in Speed control strip documented on the next page.

Rendering a Video to an Image Sequence Set

In many cases, cuting and re−arranging (editing) a codec−encoded video strip will give you fits, because of the encoding algorithm that is used internally to reconstruct each image gets 'off' by a frame or two or three. To work directly on the 'raw' frame set, a very common technique is to import your video as a strip and render it out to series of individual frames, where each frame is stored in its own image file (JPG most commonly). To do so, Add−>Movie and load your original video. Set your Format SizeX and SizeY (either to match the original, or different if you want to distort or upscale/downscale the video), set image type to JPEG, adjust your Quality settings, and in the Anim panel set your End: to the number of actual frames in the video strip. Click ANIM and a series of number files will be output to the top filespec in the Output panel. You can now delete the video strip, and Add−>Image instead, and right click on the directory name to pull in all of the images, in sequence, that are within that directory. Now, when you cut at frame 4321, for example, the next frame of the second strip will *really* start with frame 4322.

Rendering a Video from an Image Sequence Set

Ridiculously easy:

1. Add the sequence of images as described above
2. Set your Output file path and name to wherever you want to save the movie file (e.g. C:\My Documents\MyMovie)
3. Change your Format to an movie file format (AVI, MOV, FFMPEG) and CODEC
4. Set your framerate to match whatever framerate the sequence is to be played back in.
5. Set your ANIM End: to the number of images in the sequence, and
6. ANIM

The single movie file is created and saved; the name is what you specified but with the starting frame and ending frame numbers appended (e.g. MyMovie0000−0250.avi)

Previous: Manual/VSE Modes | Contents | *Next:* Manual/VSE Built−in Effects

User Manual: [Contents](#) | [Guidelines](#) | Blender Version **2.43**

Blender offers over a dozen effects that are built into Blender, and are therefore universal. Some operate on two strips; some on one, and some create a new strip. Each effect enhances your content in some way or allows professional−quality transitions.

Add Effect

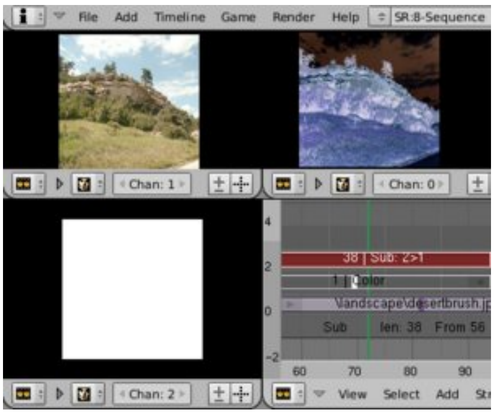
Can you hear the thunder?

The Add effect adds two colours together. Red and Cyan (Green and Blue) make White. Red and Blue make "Magenta" (i.e. Purple!). Red and Green make Yellow.

The Add Effect adds the colors of two strips together, Use this effect with a base image strip, and a modifier strip. The modifier strip is either a solid color or a black−and−whte mask, or another image entirely. The example to the right shows what happens when you add gray to an image, and animate the effect over time. The image gets bright because we are adding gray (R:.5, G:.5, B:.5) to say, a blue color (R.1, G:.1, B:.5) resulting in (R:.6, G:.6, B:1.0) which retains the original hue (relationship between the colors) but is much brighter (has a higher value). When applied to the whole image like this, the whole image seems to flash.

You can use this effect to increase the brightness of an image, or if you use a BW mask, selectively increase the brightness of certain areas of the image. The Mix node, in Add mode, does exactly the same thing as the Add sfx strip here, and is controlled the same way by feeding the Factor input.

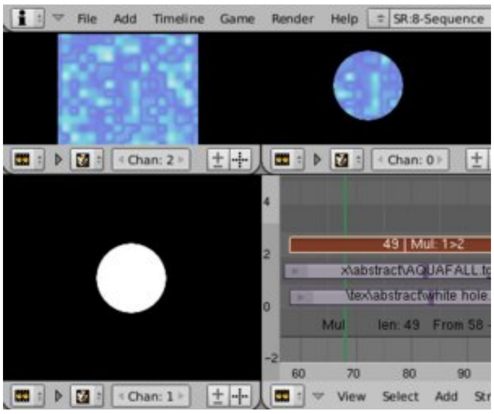
Subtract Effect



Subtract Effect

This effect takes away one strip's color from the second. Make a negative of an image using this effect, or switch the order of the strips and just darken the strip. Subtracting a hue of blue from a white image will make it yellow, since red and green make yellow.

Multiply Built−in Effect



Multiply Effect.

The Multiply effect multiplies two colours. Blender uses values between **0.0** and **1.0** for the colours, he doesn't have to normalise this operation, the multiplication of two terms between **0.0** and **1.0** always gives a result between **0.0** and **1.0** (with the 'traditional' representation with three bytes â€“ like RGB(**124**, **255**, **56**) â€“, the multiplications give far too high results â€“ like RGB(**7316**, **46410**, **1848**) â€“, that have to be 'brought back', normalised â€“ just by dividing them by **256**! â€“ to 'go back' to range of **0** to **255**â€¦).

This effect has two main usages:

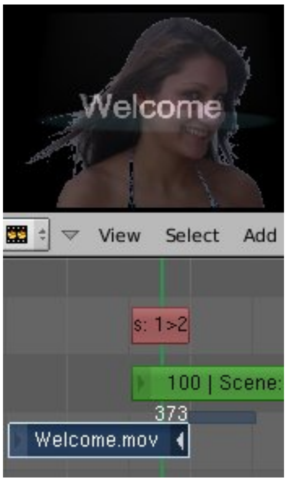
With a mask
: A mask is a B&W picture witch, after multiplication with a 'normal' image, only show this one in the white areas of the mask (everything else is black). The opening title sequence to James Bond movies, where the camera is looking down the barrel of a gun at James, is a good example of this effect.

With uniform colours
: Multiplying a colour with a 'normal' image allows you to soften some hues of this one (and so â€“ symmetrically â€“ to enhance the others). For example, if you have a brown pixel RGB(**0.50**, **0.29**, **0.05**), and you multiply it with a cyan filter (uniform colour RGB(**0.0**, **1.0**, **1.0**), you'll get a colour RGB(**0.0**, **0.29**, **0.5**). Visually, the result is to kill the reds and bring up (by 'symmetry' â€“ the real values remain unchanged!) the blues an greens. Physically, it is the same effect as shining a cyan light onto a chocolate bar. Emotionally, vegetation becomes more lush, water becomes more Caribbean and inviting, skies become friendlier.

Note

This effect reduces the global luminosity of the picture (the result will always be smaller than the smallest operand). If one of the image is all white, the result is the other picture; if one of the image is all black, the result is all black!

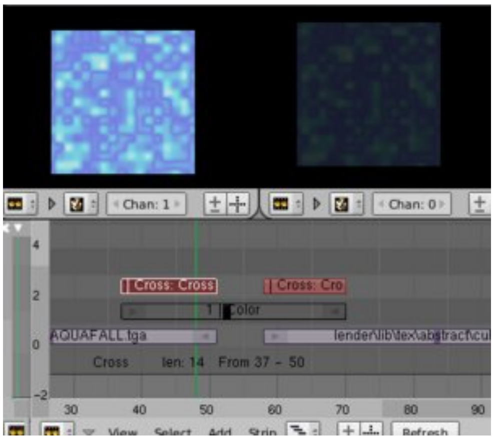
Cross and Gamma Cross



This effect fades from one strip to another, based on how many frames the two strips overlap. This is a very useful strip that blends the whole image from one to the other.

Gamma Cross uses color correction in doing the fade, resulting in a smooth transition that is easier on the eye.

Fade to Black

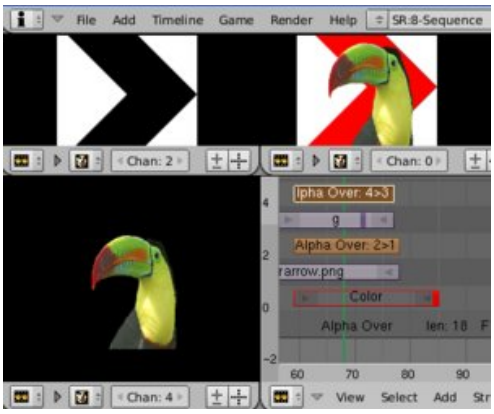


Cross−Fade between Black

Many scenes fade to black, and then fade in from black, rather than directly from one to the other.

The strip setup to do this is shown to the right. The two strips are on Channel 1, and you Add−>Color Generator strip to Channel 2, straddling the two main strips. Change the color to black, and add two Cross Effects; the first from Channel 1 to Channel 2 (black), and the second from Channel 2 to Channel 1. The first strip will fade to black, and then the second will fade in from black. Of course, you can use any transition color you want. Black is a relaxing intermediary; red is alarming. Use the dominant color in the second strip to introduce the second strip.

Alpha Over, Under, and Over Drop Built−in Effects



AlphaOver Effect

Using the alpha (transparency channel), this effect composites a result based on transparent areas of the dominant image. If you use a Scene strip, the areas of the image where there isn't anything solid are transparent; they have an alpha value of 0. If you use a movie strip, that movie has an alpha value of 1 (completely opaque).

So, you can use the Alpha Over/Alpha Under effect to composite the CGI Scene on top of your movie. The result is your model doing whatever as if it was part of the movie. The IPO Fac curve controls how much the foreground is mixed over the background, fading in the foreground on top of the background. The colors of transparent foreground image areas is ignored and does not change the color of the background.

Select two strips (Shift RMB):

- With Alpha Over, the strips are layered up in the order selected; the first strip selected is the background, and the second one goes *over* the first one selected. The Factor controls *the transparency of the foreground*, i.e. a Fac of **0.0** will only show the background, and a Fac of **1.0** will completely override the background with the foreground (except in the transparent areas of this one, of course!)
- With Alpha Under, this is the contrary: the first strip selected is the foreground, and the second one, the background. Moreover, the Factor controls *the transparency of the background*, i.e. a Fac of **0.0** will only show the foreground (the background is completely transparent), and a Fac of **1.0** will give the same results as with Alpha Over.
- Alpha Over Drop is between the two others: as with Alpha Under, the first strip selected will be the foreground, but as with Alpha Over, the Factor controls the transparency of this foreground.

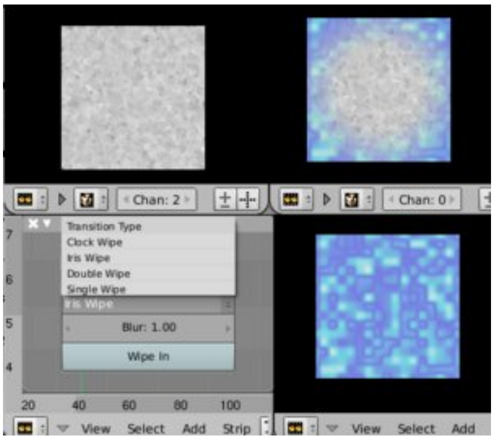
The example shows layering of AlphaOver effects. The very bottom channel is red, and an arrow is on top of that. Those two are AlphaOver to Channel 3. My favorite toucan is Channel 4, and Channel 5 alphaovers the toucan on top of the composited red arrow. The last effect added is tied to Channel 0 which will be rendered.

By clicking the PreMult Alpha button in the properties panel of the foreground strip, the Alpha values of the two strips are not multiplied or added together. Use this effect when adding a foreground strip that has a variable alpha channel (some opaque areas, some transparent, some in between) over a strip that has a flat opaque (Alpha=1.0 or greater) channel. If you notice a glow around your foreground objects, or strange transparent areas of your foreground object when using AlphaOver, enable PreMultiply. The AlphaOver Drop effect is much like the Cross, but puts preference to the top or second image, giving more of a gradual overlay effect than a blend like the Cross does. Of course, all of the Alpha effects respect the alpha (transparency) channel, whereas Cross does not.

The degree of Alpha applied, and thus color mixing, can be controlled by an IPO curve. Creating a Sine wave

could have the effect of the foreground fading in and out.

Wipe Built−in Effect



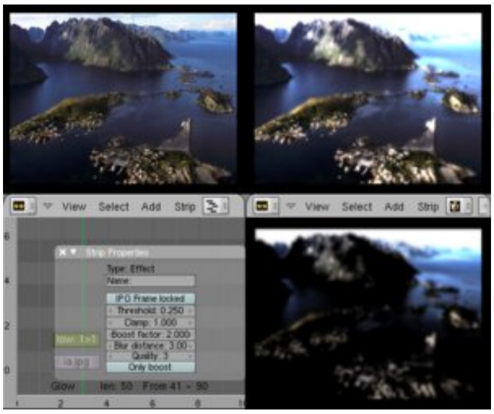
VSE Wipe Built−in Effect

Wipe transitions from one strip to another. This very flexible effect has four transition types:

- Clock: like the hands of an analog clock, it sweeps clockwise or (if Wipe In is enabled) counterclockwise from the 9:00 position. As it sweeps, it reveals the next strip.
- Iris: like the iris of a camera or eye, it reveals the next strip through an expanding (or contracting) circle. You can blur the transition, so it looks like ink bleeding through a paper.
- Double Wipe: Starts in the middle and wipes outward, revealing the next strip. It can also Wipe In, which means it starts at the outside and works its way toward the middle. You can angle and blur the wipe direction as well.
- Single Wipe: Reveals the next strip by uncovering it. Controls include an angle control so you can start at a corner or side, and blur the transition.

Note: some older plugins contain similar functionality.

Glow Built−in Effect



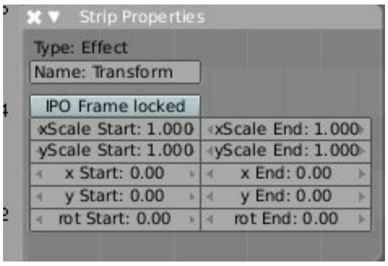
Example of a Glow effect applied to a picture.

Top left: base picture (Lofoten Islands, Norway â€“ source: wikipedia.fr);
Top right: result of the effect;
Bottom left: effect settings;
Bottom right: result with the Only boost button activated.

This effect makes parts of an image glow brighter by working on the luminance channel of an image. The Glow is the superposition of the base image and a modified version, where some areas (brighter than the Threshold:) are blurred. With the Glow strip properties, you control this Threshold:, the maximum luminosity that can be added (Clamp:), a Boost factor: for it, the size of the blur (Blur distance:), and its Quality:. The Only boost button allows you to only show/use the 'modified' version of the image, without the base one. To "animate" the glow effect, mix it with the base image using the Gamma Cross effect, crossing from the base image to the glowing one.

Transform Built−in Effect

Transform is a swiss−army knife of image manipulation. It scales, shifts, and rotates the images within a strip. The example to the right shows what can be done with a single image. To make a smooth transition to the final effect, enable the IPO Frame locked button and define a curve in the Ipo Window (Sequence mode).



With the Transform strip selected, uses the properties panel to adjust the settings of this effect:

(x,y)Scale (Start,End):
: To adjust the scale (size). xScale Start defines the start width, xScale End the end width, yScale Start the start height, and yScale End the end height. The values higher than **1.0** will scale up the picture, while values lower than **1.0** will scale it down.

(x,y) (Start,End):
: To adjust the position (shifting). x Start defines the horizontal start position, x End, the end one; positive values shift the image to the right, negative values, to the left. y Start defines the vertical start position, y End, the end one; positive values shift the picture to the top, negative values, to the bottom.

rot (Start,End):
: The rotation is in degrees (**360** for a full turn) and is counter−clockwise. To make an image spin clockwise, make the end value lower than the start one (e.g. start it at 360 and go down from there).

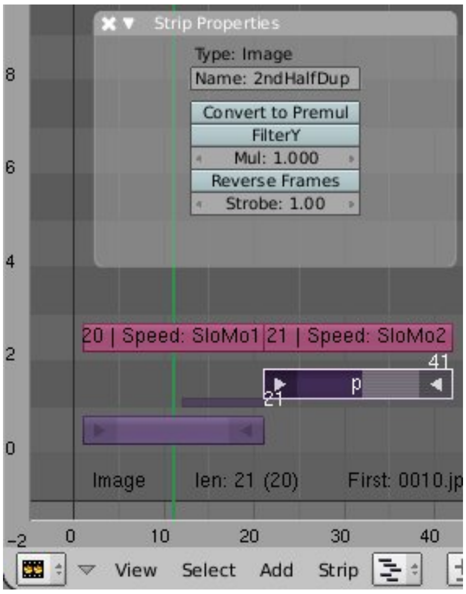
Color Generator Built−in Effect

This effect works by itself to create a color strip. By default, when it is created, it is 50 frames long, but you can extend it by grabbing and moving one of the ends. Click on the color swatch to pick a different color (by default, it is gray). Use this strip crossed with your main movie to provide a fade−in or fade−out.

Speed Control Built−in Effect

Speed Control time−warps the strip, making it play faster or slower than it normally would. A Global Speed less than 1.0 makes the strip play slower; greater than 1.0 makes it play faster. Playing faster means that some frames are skipped, and the strip will run out of frames before the end frame. When the strip runs out of frames to display, it will just keep repeating the last one; action will appear to freeze. To avoid this, position the next strip under the original at a point where you want motion to continue.

Using Speed for a Slow−Motion Effect



50% Slow motion using Speed Control

Suppose you want to sssslooow your strip dowwwwwwn. You need to affect the speed of the video clip without affecting the overall frame rate. Select the clip and Add−>Effect−>Speed Control effect strip. Click to drop it and press N to get the Properties. Set the Global Speed to be the factor by which you want to adjust the speed. To cut the displayed speed by 50%, enter 0.50. Now, a 30−frame clip will play at half speed, and thus display only the first 15 frames.

If you want the remaining frames to show in slo−mo after the first set is displayed, Kcut the strip in two, offset the second part (because the first slow−mo will actually run for more time/frames than shown), and add another Speed control as shown to the right. When the strip on Channel 1, as modified by the sfx strip on Channel 3 finishes, the strip selected on channel 2 starts, as modified by its sfx speed control in channel 3. The trick in creating the second strip is to

- Select the original strip and Shift Duplicate it. In this case, it is a 20−frame image set.
- Drag the clone to half−way above the original and drop it.

- Select the left handle (start), grab it, and move it over half−way. This changes the start frame offset; the strip will now start playing at frame 11 (relative to within itself)
- Select the right handle (end), grab it and move it over half−again as much (in this case, 10 frames) This sets the duration to be 20 frames within your video.
- Add a 50% Speed Control for the first strip, and a 100% speed control for the second. For the second strip, you are already telling blender to play 10 frames over a 20−frame duration, which is already half speed, so you don't need the speed control to slow it down anymore, just to scale the frames selected to the duration of the clip.

That's it! Set your render to animate (in this example) all 40 frames.

Why not just extend the original clip out, you ask? Well, the Speed Control operates based on the number of frames it is going to show, divided by the number of frames it has to show them over. For example, if a strip has 20 frames in it, and you specify a speed factor of 50%, it knows it has to display 10 frames. If that strip is, say, stretched to cover 30 frames of video, it will play each of the 10 frames for 3 frames (3 x 10=30). Understanding the math will help you be effective and get the control to do what you want it to.

In very simple cases it will also work to just extend the original clip. This is a very nice feature intended for story boarding and for filling gaps. Add a speed control, ignore the IPOs and all the other parameters and just extend the right handle of the original clip. Remember that this is only possible, if the underlying clip ends at the right handle position. (If this isn't the case because you used the knife tool to create it, add a meta for the original clip first.).

Using speed control for frame matching

To get even finer control over your clip timing, you can use IPOs! This can be done in several ways.

- In the default configuration, your IPO curve simply map input to output frame scaled down to the range of [0−100] [0−1] in IPO curve coordinates. This is enough, if you want to add simple flickering effects.

The speed effect can do even more for you. You can do actual frame matching! That means: you have a certain position at which a certain frame should get displayed (maybe when matching your video track to an audio track) and blender will adjust the speed of the clip smoothly to make this happen.

- You can disable scaling in X direction by clicking on "IPO−frame locking". That means: input frame numbers are identical to IPO curve X−coordinates.
- You can disable scaling in Y direction by disabling "Scaling [0−1]". That means: output frame numbers are identical to the IPO curve Y−coordinates.

After that, use the N−keys window fo the IPO−window to place your control points in a frame exact way.

Changing video frame rates

You can use the speed control to change the frames per second (fps), or framerate, of a video. If you are rendering your video to a sequence set, you can effectively increase or decrease the number of individual image files created, by using a Global Speed value less than or greater than one, respectively. For example, if you captured a five−minute video at 30 fps and wanted to transfer that to film, which runs at 24 fps, you would enter a Global Speed of 30/24, or 1.25 (and Enable Frame Blending to give that film blur feel). Instead of producing 5*60*30=9000 frames, Blender would produce 9000/1.25=7200=5*60*24 frames. In this case, you set a Sta:1 and End:7200, set your Format output to Jpeg, 30fps, and image files 0001.jpg through 7200.jpg would be rendered out, but those images 'cover' the entire 9000 frames. The image file 7200.jpg is

the same a frame 9000. When you read those images back into your film .blend at 24 fps, the strip will last exactly 5 minutes.

Previous: Manual/Using VSE	Contents	*Next:* Manual/Sequence Plugins

User Manual: Contents | Guidelines | Blender Version **2.43**

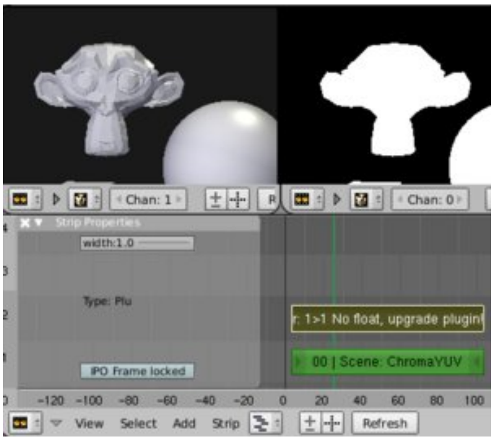
Plug−ins are just as easy to use as built−ins; they just require an extra step in finding them on your system and selecting the one you want based on its filename. By default, they are located in a Plugins directory within your Blender installation directory. In a default Windows installation, it is "C:/Program Files/Blender Foundation/Blender/plugins/", or wherever the path is entered within the User Preferences File Paths window.

You find these DLLs by:

- Visit the Blender Resource Page for Plugins
- Hunt the internet and visiting programmer's websites
- Explore the plugins that are installed with Blender.

When you find them, download the version that is appropriate for your operating system (there are different flavors for Unix, Windows, and OS X), saving them in the Blender Plugin Sequence directory. The location of this directory on your machine is under your install directory or overridden by a setting in your User Preferences panel.

Alpha Matte Plugin Effect

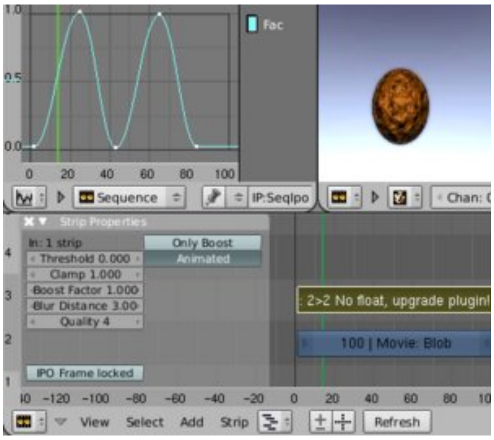


VSE AlphaMatte Plugin

The Alpha Matte plugin converts the alpha channel of a strip to a black and white image corresponding to the alpha values; transparent is alpha 0 which is shown as black. Use this matte as a garbage matte, or as a multiplier against the selected strip so that a background strip will show through.

Note: Only your Scene and certain image types (PNG, EXR) support an alpha channel.

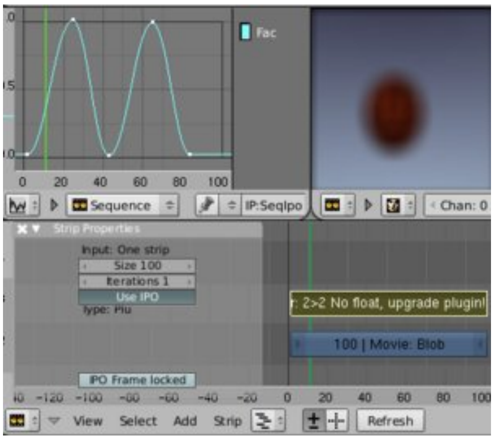
Animated Glow Plugin Effect



VSE AnimGlow Plugin

The Animated Glow plugin selectively increases the brightness of an image. Pixels with a brightness below the Threshold are not affected, and those selected will not be brightened above a Clamp value. The amount of glow is determined by the BoostFactor, and the Distance the glow extends outward is also selectable, as well as the Quality. In addition, you can see the composite glow, or just the amount of glow Boost that is added. This example shows a glow in process at about 50% of a 100% Boost Factor; the image will glow twice over 100 frames, simulating a light being turned up and down.

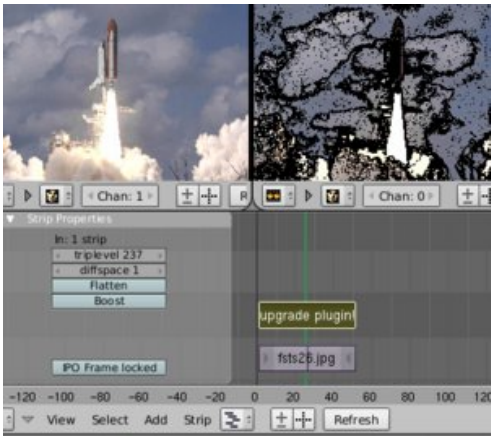
Box Blur Plugin Effect



VSE BoxBlur Plugin

The BoxBlur plugin blurs the pixels in an image by the number of pixels defined by Size. The smoothness of the blur is increased by increasing Iterations. The blur can be animated over time by enabling Use IPO and creating an animation curve as shown. This example shows a blur in process at about 25% of a 100 pixels; the image will go in and out of blur twice over 100 frames, simulating a camera focusing.

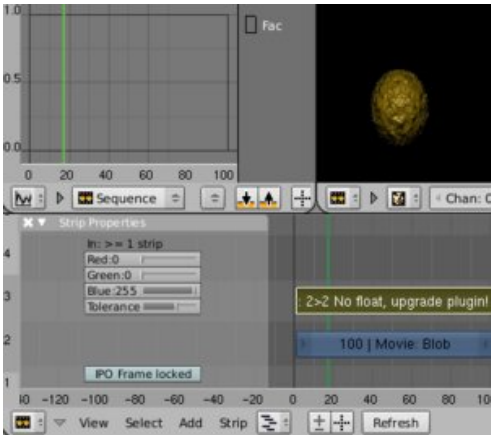
Cartoon Plugin Effect



VSE Cartoon Plugin

The Cartoon plugin pixelizes and flattens the colors in an image, making a "real" image into something that would be drawn in a comic book. The trip level and difference space determines where the black "lines" should be drawn. Flatten and/or Boost image colors to add false color.

Chromakey RGB Plugin Effect



VSE Chromakey RGB Plugin.

This plugin turns the pixels which have the color determined by the RGB sliders, within an error Tolerance, transparent. The example above takes our blob on a shaded blue background and treats the background as a "blue screen", turning it all transparent. The same could have been done for green, which is called a "green screen". This allows us to now overlay the foreground blob over a new background using the builtin AlphaOver effect. This effect is not animatable.

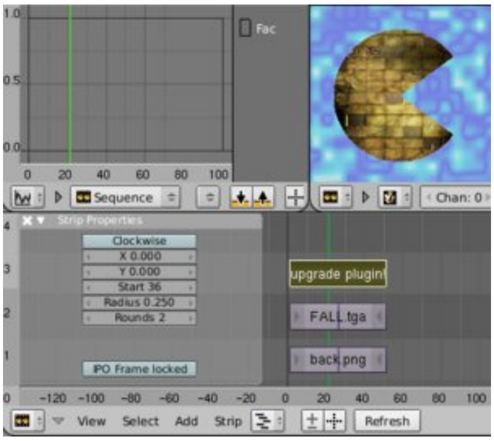
Chromakey YUV Plugin Effect

VSE Chromakey YUV Plugin.

Based on the luminance (Y) and chromiance (U and V) values of a foreground strip, this plugin shows the pixels of a background strip underneath the foreground. This plugin turns the pixels which have the color determined by the YUV value sliders, within an error tolerance defined by TolYUV sliders, transparent. The amount of blending is controlled by the Blend slider. The example above takes a swirly foreground image and puts our blob on a shaded blue background as the background. It treats certain colors within the foreground as the "blue screen".

YUV chromakeying is often used when an actor is filmed in front of a black screen, and the YUV keying turns the black background to alpha, letting the background show through.

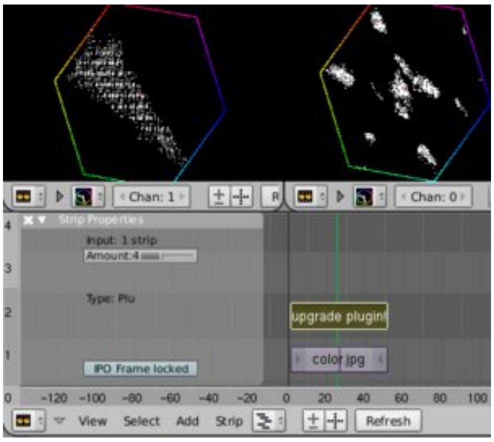
Clockwipe Plugin Effect



VSE Clockwipe Plugin

Remember the old countdown header in black and white movies? Based on the Starting angle, this plugin reveals the background image by wiping in a clockwise or counterclockwise direction. The Radius is how much of the image to reveal, and the clock goes round and round the number of times specified. As it completes one round, it reverts to the foreground image. Use this effect as an inventive wipe between scenes (using a radius of 0.5).

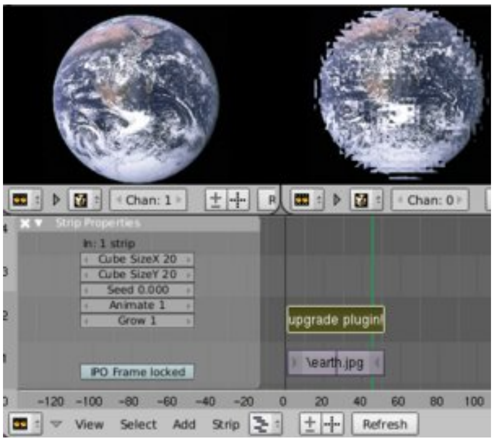
Desaturate Plugin Effect



VSE Desaturate Plugin

This interesting effect removes the saturation from an image starting from the middle and working its way outward. The net effect is that for middle−range values, the colors in the image are made extreme. A saturation of zero is a black and white image, but in the middle, colors above the threshold become oversaturated (posterized). Notice that here we use the Chroma Vectorscope to see the color saturation map.

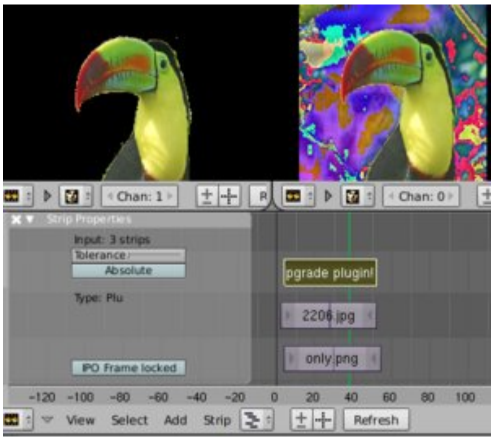
Dice Plugin Effect



VSE Dice Plugin

Better than a Ginsu knife, this plugin slices and dices your image into cubes, and flips them. When animated and allowed to Grow (equal to 1), the effect grows over time to the cube size specified. A very interesting wipe effect, and you will never need a bandaid.

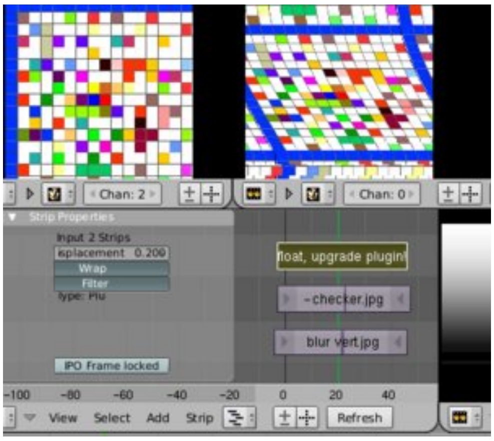
Diff Plugin Effect



VSE Difference Plugin

The Difference plugin subtracts one image from another, and shows you any difference between them, either as a color map (using color math) or as an Absolute black and white mask. If there is no difference, it passes through the original. Can you figure out what image we compared the Toucan to? In this example I played a little trick on you. I compared the image of the toucan only to the image of toucan in its environment. As a result, the environment is a color negative, while the toucan is the toucan since there is no difference in the toucan area between the two images. Gotcha!

Displace Plugin Effect

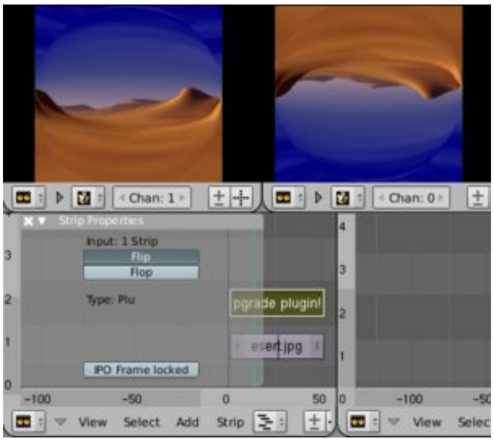


VSE Displace Plugin

First selecting a vector map (a black and white image that tells the plugin what to do) and an image to manipulate, the Displace plugin shifts the pixels sideways and vertically. Use Wrap to make a wrap−around transition so that pixels shifted off the image on one side are used to fill in the other side. Filter does some anti−aliasing, making the image smoother.

A positive displacement shifts the pixels down and to the right, while a negative displacement shifts them up. Partially shown in the image to the right is the vector mask used for this example: a simple gradient that goes vertically from black to white. The shade of gray is multiplied by the displacement to determine how far to shift the pixels.

FlipFlop and FlipIt Plugin Effects



VSE FlipFlop Plugin

Flip an image upside down, or Flop it sideways left−right. Simple and cute.

FlipIt adds the option to rotate the image 90 degrees.

GreenScreen Plugin Effect

VSE GreenScreen Plugin

Combining live action footage, filmed in front of a green screen, and composited over a background, is simple and fast thanks to this amazing new (Feb 2007) plugin from Paprmh, proving once again that the Sequence plugins are alive and well.

Simple to use? Just load your greenscreen movie and load your backaground strip. Shift−select the greenscreen, then the background, and add this effect. When you click the check/ibuf2 button twice (until it turns yellow), the plugin automatically substitutes your background for the green removed. While nodes may be more flexible, they are about 15 times slower; using this plugin you can process a ten−minute run−time green−screen footage clip in a half an hour (on my Centrino(r) laptop).

This easy−to−use plugin takes out the green with values between the Ymin and Ymax values, and leaves U

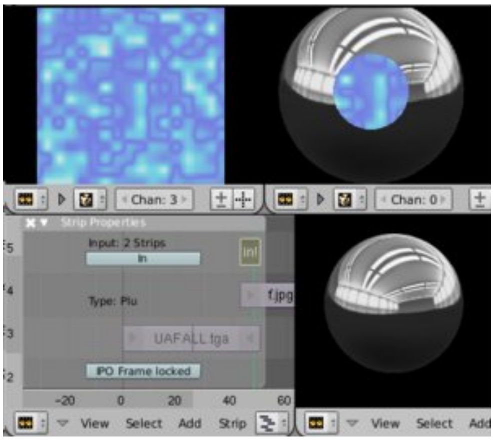
(blue) and V (red) alone based on clipping values set with the sliders. It also has special filters for DV 4:1:1 footage that smooths out the color channels. The variable color spill factor, spill/+lum button get rid of that dark outline and any green that may have spilled onto the actors. The edge blur melds (anti−aliases) the edges of the keyed image to the background.

Download this plugin and visit his site for a complete explanation and tutorial that even covers garbage matte techniques using the Sequence Editor.

Channel 0 outputs are selected through a display control. Choices include an the image with green as alpha 0.0 (using the black swatch), colored backgrounds using the color swatch, or UV masks (display:1), alpha masks (display:2), or YUV (display 3,4,5 respectively) using the display selector.

If you can't afford a $400 muslim greenscreen, rumor has it some Blenderheads have even had fair results using a $4 tablecloth.

Iris Plugin Effect



VSE Iris Plugin

This transition effect reveals the second strip while wiping the first by a decreasing circle, like the closing of an iris. Selecting In covers up the base image by revealing the second as the circle (iris) closes.

Jitter Plugin Effect

VSE Jitter Plugin

Too much caffeine? Can't hold the camera steady? Need to simulate Brownian motion? How about being time−phased out of sync by a harmonic distortion in the time−space continuum? Yeah, that sort of stuff happens to me every day too. But no one believes me, so use the Plugin to show them what it's like. Set the X and Y to be in pixels, and enable Random.

Lightsaber Plugin Effect

Lightsaber Animated Plugin

From: Intergalactic Indemnity Assurance Corporation

> To: Claimant
> Re: Request for Disability
> Date: 15−05−3043 Terra−GMT

Dear Sir/Madam:

Our examiners have carefully reviewed the photographic evidence you submitted in connection with an unfortunate light−saber incident you claim happened during the filming of "Elephants Jedi" which resulted in the loss of your arm. After careful inspection, we have concluded that the so−called "lightsaber" was in fact a special effect which you created and staged to look like an accident waiting to happen. Your claim has been turned over to the authorities for prosecution.

Sincerely,

> Jackaky Skinflint, CPA, AFP, EXT, CLIM, GAK
> Claim Inspector #4395, Intergalactic Indemnity Assurance Corp.

WARNING: This message is confidential and for its intended recipient only. If you have received this message in error, please report to your local health office and submit to a free brain wipe. Thank you.

This effect does an edge detect on the base image, and adds a glow to the edge. In the example above, the scene has just a tube. The plugin makes that tube into a lightsaber, and the effect strip is AlphaOvered the live−action footage. Animating the tube to track his hand would make it look like he's about to lop my arm off. At least, that was the idea. Too bad Jackaky wasn't fooled.

LSD Plugin Effect

VSE LSD Plugin

Now you can experience the joy of mind−altering drugs without actually incurring any risk or harm, and saving a lot of money in the process. This self−animating plugin does bizzarre things to the colors of your image. There is not control over this effect; it does what it wants to (just like the real LSD). But, no flashbacks (unless you duplicate the source strip and effect).

OldMovie Plugin Effect

VSE Old Movie Plugin

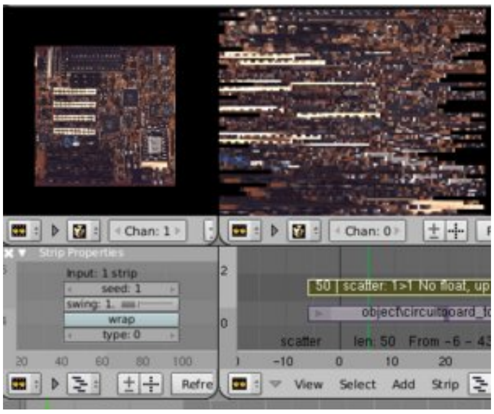
Back in the old dayz (I just love saying that), of black and white movies, the film stock was grainy, and the developer brushes would clog and scratch the negatives, resulting in long black streaks on the print. Real crap for quality. If you long for the old days, or want to post some Lost tapes from the original Dharma Initiative, use this effect to make them seem old. Thank you, and Good Luck. Oh, and if you do a good job, everyone will want to watch it again.

Red−Green Stereo Plugin Effect

VSE Red−Green Stereoscopic Viewer Plugin

Don't throw those Trix glasses away! Use two images, one taken from the persective of the left eye, another from the right, as the two images to feed this plugin. The Plugin generates a red−green stereoscopic image that appears 3−D when you put the glasses on. If you don't have a pair, you might be able to borrow indefinately (steal) your little sister's from one of her DVDs, or find a site on the web that gives them away.

Scatter Plugin Effect



VSE Scatter Animated Plugin

So there we were, outbound from AlphaCentauri, when our Falx Confibulator board blew. We pulled the circuit board, and our logic tracer was showing us a problem somewhere in the VSE circuitry. Drifting just outside the nebulae, way too close to *the Fringe* for any of us to feel comfortable, we tried desparately to get Tech on−line to give us some assistance. "Hello, Tech? We have a problem here. Can you see the board?" we asked. "Barely" came the wiz's resonse, "I've got another one here and I'll walk you through..zzzcchdhitchchch" and the signal got garbled and full of static all of a sudden. "Must be an ion storm inside the nebula" my comms officer surmised. "Dammit, we need a clear signal! I can't tell what chip he's pointing to, and unless we get this Confibulator back on−line, we're stranded out here!" I yelled. With that, we all turned to the viewport, and watched in quiet desparation as we drifted ever closer to *the Fringe*.

Previous: Manual/VSE Built−in Effects	Contents	*Next:* Manual/Working with VSE

User Manual: Contents | Guidelines | Blender Version **2.43**

Working with Blender for Video Editing

This section covers the workflow of the video editor, providing numerous examples of practically using the tools described in the previous sections. While editing can occur in any order, in the real world it begins based on what footage has been shot, called "in the can".

Lots of times, footage is shot out of sequence. For example, they may shoot the fight scene first, then the introduction, then the thrilling climax, and titles added just before release. This style of shooting is motivational, and you have some exciting dailies to show the producer and to lock in total funding, but is hard on the actors to give a good performance, and hard on the editor because he can't tell the whole story. It's analogous to writing a book by starting to write chapter 13 first.

This section assumes that all footage has been shot, and we want to edit the video in sequence to produce a post−production pre−audio copy.

This section also assumes you will be producing an animation, and that the audio track will be recorded or synched after you are done, so you don't have to make markers or insert synch frames into your video.

Cut Rollup/Rolldown

The Director says "lights, camera, action" to turn on the lights, start the camera rolling, and to get the actors in gear to do their thang. In between camera and action, a clapper claps to enable us to sync audio later with the video. After the scene is complete, he says "cut" which stops the action and camera. When you get the raw footage uploaded, you need to chunk it out into clips that can be reviewed, and the best chosen. The part where the camera starts up to the clap is the rollup and is useless. Any filming from the time he says cut to the end of filming is rolldown garbage.

Everything from the start of the raw footage up to the clapper making the clap can be cut. To cut rollup, simply position your cursor to the frame where the clapper bar meets the clapper as seen in the Sequence window set on Image display (the strip must be selected). Back up one frame rightarrow and press K to cut, and confirm. The strip is cut in two; just like using a bench and a razor. Select the rollup and press X to delete it. Grab and slide the remainder to frame 1.

At the end, everything filmed after and including the word "cut" can be cut, leaving a middle segment that may contain multiple partial or complete takes. To cut rolldown, find where he says "cut", and Kut and X delete the tail (unless there's another scene and/or take left). Sometimes the director will forget to say cut, and the camera keeps rolling, so you have to cut it for him by looking at it and seeing when the actors finish. Cut from there to the next clap (or end of footage), but move up the footage to a higher channel. It will be kept for review by the director, but won't be shown since the video beneath it will be shown instead.

Cut out Flubs

For example, an actor may get halfway through their lines, flub the last word of the third sentence, do an "ahblahyadayada" to get the marbles out of their mouth, and start over with the start of the third sentence without prompting. In this case, the middle flub needs to be cut out of the middle (more often than not, they will have moved their head or background has changed resulting in a discontinuity, and the footage will have to be tossed. However, using the offset and displace nodes and transform sfx, it may be possible to salvage it and splice the two). Cutting without a clap is a little dangerous, since the audio engineer is going to have to cut out the exact same flub. In this case, keep a record the actual number of frames cut by moving the cut to a

higher channel, and the audio engineer can calculate how many milliseconds to cut as well.

For this reason, when you cut the flub, just move the flub up a channel or two, and slide the remainder underneath it. This way the flub is still available for a gag reel, and is available for you to review when the audio engineer asks how many frames were cut because he lost the tracking sheet.

You may also cut out dead time in this manner; cutting straight to the action (from the clap to the start of useable footage) while preserving frames for later review or use by other people.

Save Scrubs

I hope the previous sections show you how you can use Blender as a full featured video editing system. Ok, time to save:

- In the Buttons window, Output panel, change the output filename to reflect the scene and take number (shown on the clapper). You may want to disable extensions.
- In the Timeline window header, set the start and end frames for the take
- Scrub it
- Enable Do Sequence in the Anim panel, and set whatever your format is needed
- Click Anim and that clip will be saved.

Let's move on to using Blender to composite the action footage into a movie.

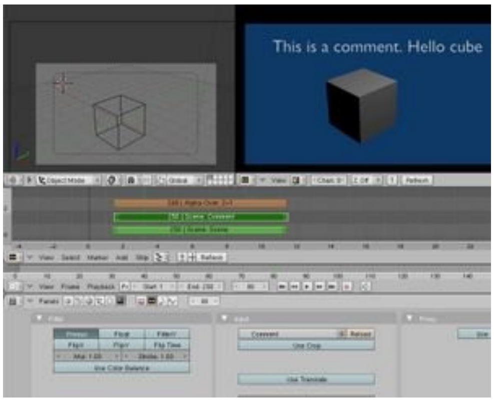
Video Conversion

Many times input video will be in different formats (avi jpeg and quicktime) and/or different sizes (resolutions) and/or different frame rates. As we have seen, Blender automatically converts stills (images and image sequences) into video when Added. Regardless of the input, Blender renders a single output in the format (codec, resolution, framerate) specified. This means that you can Add−>Movie and pick a QuickTime movie and load it into the VSE. If you specify "Avi Jpeg" as your output format (in the Buttons window, Scene Render context, Format panel), and "Do Sequence", Blender will convert the QuickTime video to a compressed .avi file which can be played in Windows Media Player.

Images or video are automatically scaled or cropped to fit within the resolution window specified in that output format panel. If they are smaller than the output resolution, they are centered in the video (unless offset by the Transform effect).

Video of differing frame rates can easily be mixed and matched. With regard to frame rate, video is imported on frame−by−frame basis based on its run−time. If a 10 second video that was encoded at 24 frames per second is loaded, 240 frames will be loaded in, even if your output is, say, 30 frames per second. This means that your exported video sequence may run faster or slower than input. In this example, the imported video will be export to run for a duration of 8 seconds (240 frames / 30 frames per second = 8 seconds) instead of 10 seconds. If you want to correct this, select the segment and use the Speed Control effect, specifying a factor of 24 / 30 = 0.80, and the segement will be run in slight slow motion relative to the 30 fps you wanted. You will have to extend the video strip to end two seconds later (to fill up the gap) by right−clicking and grabbing the end frame, and dragging it to the right.

Overlay Text



In addition to being able to stamp common tracking information on the render, you may want to overlay some comments on top as well, such as a title, or questions for the director on animation, or to call attention to a material or lighting choice.

To do so, create a new scene and in a 3D window, switch to camera view. In that view, add a text object, and edit it to be your comment or import a text file. Position it roughly where you want it. In the VSE, add that comment scene as a scene strip, and AlphaOver it on top of the base video. Use the Premultiply option on the Comment strip so that it does not affect the colors of the base image.

Previous: Manual/Sequence Plugins	Contents	*Next:* Manual/Video Sequence FileOrg

User Manual: Contents | Guidelines | Blender Version **2.43**

Collecting and Organizing Footage

Video is huge and even a screaming machine can take a half−hour to encode a three−minute segment. Accidently deleting or overwriting a source file is disasterous, as there is no way to go back in time and reshoot a live action scene. There are many ways to set up your file structure, so this topic is just a suggestion.

Gimme Some Room

When organizing your files for a major video project:

- Check with your LAN administrator and watch your disk space usage, ordering additional disks before you crash the server or workstation. Each 1 hr DV tape takes 20G of disk space; just the raw video. Seven hours of shooting is 140G, and if taping a five day class, you're looking at a terabyte just for the raw video (no clips extracted) and audio files.
- Create a non−destructive structure and process where it is virtually impossible to accidentally overwrite a source file
- Buy extra backup tapes; you'll need them.

Media Directory Structure

Here is a suggested file/directory organization chart, just to get you started thinking about how best to organize your files. In concept, you have

- Standards that you want to adhere to, such as procedures for uploading video, or settings you use in generating video clips so that they are all consistent.
- Library of media you can re−use from project to project
- Project−specific media, shot for a specific purpose.

As you work through a project, keep an eye out for things to add to the Library. As you discover new tools to use, add their identification and use to Standards. When you start a new project, create a folder for that project:

C:\Media − all media goes here
C:\Media\Standards − all your checklists, procedures, settings
C:\Media\Library − anything you might be able to re−use
C:\Media\Library\Stock − stock photos and clips you license. Create a subfolder by source, and then subfolders under that by topic. In the subfolder that has the source, put a copy of the license agreement
C:\Media\Library\Public − public domain images and sequences and textures
C:\Media\Library\Corporate − logos, powerpoint templates, letterhead, building pan, show−off area pan, interior corporate shots
C:\Media\Library\People − head shots of employees, clips of them acting. Create a subfolder for each person. Include in that subfolder a scanned copy of their signed release.
C:\Media\Library\Clips− clips shot and owned by the company, salvaged from prior projects. Create subfolders based on topic (Training, Ads) and the subtopics (each class, each ad campaign)

C:\Media\Projects\xxxx\

The project−specific area is where you will be spending much of your time and energy; create a folder, where xxxx is the project name. Under that folder, I suggest you create these folders:

\PreP − Pre−Production concept development, scripts, budget, storyboards, timeline, casting call, notes go here. Most importantly, a shooting script goes here that lays out each scene and possible shots to take.

Scene−Shot−Take

Each *scene* tells an important part of the story, and may only be a few minutes. The director may shoot that same scene from different angles, under different lighting, or just give different direction to the actors. For each *shot*, you may have to film multiple *takes* until the actors get their lines right.

\Prod − Production starts when principal photography starts. All raw footage, as well as extracted, acceptable, clean takes are put here, from all the stuff filmed. These clips are cleaned and scrubbed, samples, video works in progress go here. Your input tothe VSE comes from here as you work.

\raw − raw video uploaded from videotaping. Files named by date they were shot, with spreadsheets to index their contents
\clips − useful clips extracted from raw video, named by Scene−Take−Version−Description, such as "05−19a−bad lines.mov" for Scene 5, Take 19, original version (a). Good clips are named with the word "good" in the filename.

Dailies

Finding good files for a specific date is thus easy, and called a "daily" because the editor reviews each day's shooting ("viewing the dailies") to see the good stuff coming in from location. The bad stuff is still clipped and saved for the gag reel.
Good clips can then be colorized and processed using Blender nodes, resulting in a new version of that clip.

\stills − In here are stills taken on−set, or to be used as backdrops and matte paintings.
\CG − guess what goes here−your .blend files! Name your files by act−scene−take or just scene−take−name, such as "07−04−Emo flips out.blend"; from project Orange, the world's first Open movie, and the first European HD release, named their files "act−scene−description.blend"; there were 8 acts and about 20−30 scenes in each act.
\CG\render − your intermediate renders and output from rendering, compositing. When you are ready to share, copy the .mov or .avi to the \clips folder.
\comp − This is the CG equivalent of the "clips" folder, where the output of the VSE goes when used to composite footage rendered out in layers.

Levels of Compositing

There are two levels of compositing:

- Process multiple renderlayers to get a CG strip (put in the "\comp" folder).
- Process the background, live action (green screen removed), and CG together to get a final strip (put in the Post−processed folder). Use the VSE within the CG directory to process the different renderlayers into a final CG product. Also use the VSE to produce the final composite; the VSE does either one well. The final composite in put into the "comp" directory, and may be formed from many layers:
- Add the background image as a strip to a channel, usually channel 0
- Add processed live background action
- Add CG scene or sequence or movie output for CG elements on top of background

- Add processed live action that goes in front on CG elements
- Add blending CG elemements and overlays like lighting, fire, mist, smoke dirt, fog, watermarks

  \post − Post−Processing − the composited footage for each scene, as well as final cuts, sequenced using the VSE, go here. You may produce, for your project, three cuts (the PG cut, the R cut, the Director's cut, etc). Also here are CD sleeves, poster art, cover art, silkscreening masters; everything needed to master the final product

Files in the "\post" are spliced using the VSE. A single .blend file can hold the whole movie, and should be named based on what it produces. The editor may produce several versions of the entire movie, making a linear cut and a flashback cut; an "R" rated cut and a "PG" cut; a commercial release and an extended edition cut. Each of these cuts would be placed in the "\post" folder, along with the VSE .blend files used to produce them.

Previous: Manual/Working with VSE	Contents	*Next:* Manual/VSE Examples

User Manual: Contents | Guidelines | Blender Version **2.43**

This section provides some practical uses for Blender's Video Sequence Editor (VSE).

Making a Slide Show

Allright, you've just come back from vacation at the beach and before your beautiful tan starts peeling, you want to make a video/DVD showing the pictures you took. Simple:

- Add−>Images and (the VSE window changes to a file browser) right−click each of the pictures (probably jpeg) that you uploaded from your digital camera and want to include. If you want to include a mass of them, just hold down the right button and drag the mouse over their filenames.
- When they are all selected, click the Select Images button and (the file browser window changes back to the VSE) drag and drop the strip to start at frame 1
- Set your end frame to the end frame of your strip (the number of images selected). Hint: use the Channel 0 "Last:" information
- In the Scene−Render buttons, set your Frames/Second to 1 (format panel) and choose an .avi or .mov format. This will generate a video that shows each image for 1 second.
    - Alternatively, you can use the Speed control effect on the strip to slow it down even more.
    - If you want the slides to play longer than that, or want them to cross over, you may press Y to separate the image strip into individual image sequences; a popup dialog will ask you how many frames to make each image (at 25 fps, enter 125 for a five second showing of each image, for example). You can then arrange and re−arrange the individual slides however you want.
- Enter your output filename (output panel), enable Do Sequence, and click Anim.

I'd really like to make it more complicated than that, but sorry, it is that easy. Your resulting video will play your pictures, one every second.

Converting a Frame Sequence to a Movie Clip

Suppose you have (wisely) rendered your animation to a frame set sequence, such as a sequence of PNG or EXR formatted image files. The files are named MyAnim−0001.exr through MyAnim−0030.exr. In your working VSE, choose Add−>Sequence. Your VSE pane changes to a file browser window. Use that window to navigate to the directory where your images are stored, RMB click and drag over all 30 filenames. If you miss one, you can just RMB click it and the one previously selected will remain selected. Upon clicking "Select Images" your screen will return to being a VSE, and you mouse will have a purple horizontal strip attached to it. The strip represents your image sequence, and has a number in front of it (the start frame) and a number behind (the last frame). Move your mouse to position the strip so that it has a start frame of 1, and LMB click to drop it. Refer to the reference manual for more possible mouse and hotkey actions on the strip.

In the Timeline header, set the End: frame to be the last frame of the strip. Click the VCR−like play button to see the animation preview in the VSE Preview window pane. It will repeat forever, so click the pause button when you've seen it enough times (Play turns into the Pause button, so you don't even have to move your mouse.)

In the Buttons window, Scene (F10) buttons, Render buttons, set (going left to right):

- In the Output panel, set the output render directory to the directory and name you want for your avi movie
- In the Anim panel, click the "Do Sequence" button.

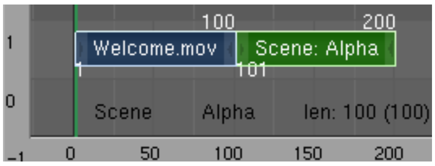
* Lastly, in the Format panel, select AVI Codec and [Manual/xx|choose your codec].

Cross your fingers and click Anim. A popup Render window will show each frame of your movie, pretty quickly, and Blender will create the file named in your Output Render field with the .avi extension. You can now play the video in your favorite media player.

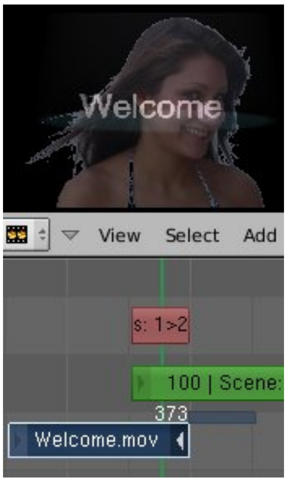
This simple example shows you how one screen layout has all windows needed in your workspace, and how they work together to get the job done.

Splicing and Transitions

Splicing is making something by taping together parts that have sliced. In video, there are many ways to join up parts or transition from one to the other. Almost all of these techniques originated with physical film or inventive darkroom techniques, so as you read them, try to imagine how you might do these with physical film.



A cut splice is simply the two pieces of film strips taped finish−to−start.



Making the two strips transition from one the other is accomplished by overlaying the srips and adding a Cross or Gamma Cross or Wipe effect.

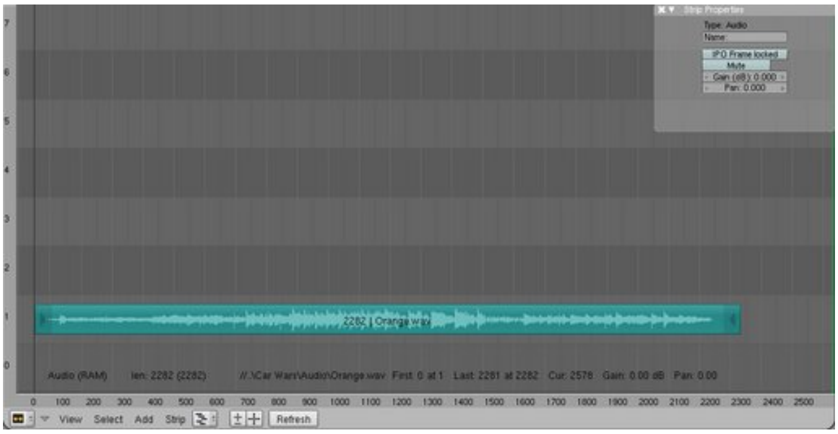
Fading in or out is crossing a color generator effects strip (set the color in its properties panel) with the video strip. Usually you fade to or from black.

Note that the Alpha effects do not work with the color generators; you can only Cross with them.

Previous: Manual/Video Sequence FileOrg	Contents	*Next:* Manual/Audio_Sequences

User Manual: Contents | Guidelines | Blender Version **2.42**

Sound Sequence Editor



A sound strip in the sequence editor.

Blender contains a multi−track Audio sequencing toolbox. You can add WAV files from your hard disk as a file, or as encoded within a movie, and mix them using an IPO Curve as a volument control.

Options

Mode: Video Sequence Editor

Hotkey: Shift A

Menu: Add−>Audio (RAM) or Add−>Audio HD

Audio−RAM loads a file into memory and plays it from there. You can only load stand−alone WAV files. Audio−HD plays the sound back from the hard disk and thus does not take up memory. With Audio HD, you can load stand−alone WAV files, but also audio tracks from movies.

For either, a green audio strip will be created. With Audio RAM, a waveform is created that shows you the waveform inside the green strip, scaled to the height of the green strip. Since Audio−RAM files are read into memory, changing the audio file will not affect playback, and you will have to re−open the file so that Blender re−reads the file.

Hiss, Crackle and Pop

Some audiophile users report that Hiss is introduced sometimes if Audio RAM is used. There must be some decoding or sampling going on, that does not occur when Audio HD is used, that introduces some playback noise. If you hear pops and crackles, usually that is a sign that your hardware cannot keep up in real−time playback. They will not be present in your final rendered animation output (but they may show up in Game mode.

Audio Mixing in the VSE

You can have as many Audio strips as you wish and the result will be the mixing of all of them. You can give each strip its own name and Gain (in dB) via the N menu. This also let you set a strip to mute or 'Pan' it; −1 is hard left, +1 is hard right, with percentages in−between.

Overlapping strips are automatically mixed down during ANIM processing. For example, you can have the announcer on channel 5, background music on channel 6, and foley sound effects on channel 7.

Working with Audio Tracks

An audio track (strip) is just like any other strip in the VSE. You can grab and move it, adjust its starting offset using RMB over the arrow end handles, and K cut it into pieces. A useful example is cutting out the "um's" and dead voice time.

Animating Audio Track Properties

A 'Volume' IPO curve can be added for the selected audio strip by Ctrl LMB clicking in the IPO Window as it is done for effect strips. The Fac channel controls the volume. You want to set a value somewhere between 0.0 and 1.0, and the volume becomes that percent; 0.6 is 60%. You can add a gain to the volume through the strip properties (N). You can make a curve by having multiple points, to vary the volume over its length. Press TAB to edit the curve, just like any old bezier IPO curve.

IPO frames 1−100 correspond to the whole sample length. In the Y direction, 1.0 is full volume, 0.0 is completely silent. To make IPO frames match video frames, switch on "IPO frame locking" in the N−keys dialog. Only the FFMPEG−output system is currently able to mix audio and video into one output stream. Use Ctrl LMB to add IPO control points, and tab to edit an IPO curve.

Animating an audio strip affects the volume of the strip in the resulting composite. Use IPO animation on an audio strip to fade in/out background music or to adjust volume levels. Layered/crossed audio strips are added together; the lower channel does not override and cut out higher channels. This makes Blender an audio mixer. By adding audio tracks and using the IPO curves to adjust each tracks' sound level, you have an automated dynamic multi−track audio mixer!

The output is therefore a video file if the ANIM button in the Anim Panel of the Scene Context/Render Sub−context is used as described before. An audio file may be created via the MIXDOWN button in the Sequencer button of the Scene Context, Sound Sub−context. This WAV file contains the full audio sequence and is created in the same directory of the video file and with the same name but with a .WAV extension. You can mix Video and Audio later on with an external program or by adding it to, for example, an image sequence strip as described above.

The advantage of using Blender's sequence editor lies in the easier synchronization attainable by sequencing frames and sound in the same application.

To enable audio synchronisation after importing an audio track, select the Scene button (F10) in the buttons window then choose the Sound Block Button (small blue sine wave). In here you'll see the Sync and Scrub tools.

- Sync lets Blender drop image frames to keep up with realtime audio when you play an animation in the 3D window. This gives you a rough overview of the timing of your animation.
- Scrub allows you to drag your frame−marker or change frames in any window and it will play a clip

of audio for that point in time.

Draging the frame−marker over a range of frames in the Action Editor will allow you to hear roughly where specific sounds occur so that you can key poses or shapes on this frame.

Previous: Manual/VSE Examples	Contents	*Next:* Manual/Extending Blender

User Manual: Contents | Guidelines | Blender Version **2.45**

Introduction

Unlike many programs you may be familiar with, Blender is not monolithic and static. You can extend its functionalities without having to modify the source and recompile. There are two ways of extending Blender: Python scripting and Binary Plugins.

Outside of the Blender executable, there are literally hundreds of scripts written by many people, and many python script resources available within this wiki. While many of these scripts cannot be included within the Blender release for logistical or support reasons, many of them work reliably and are very useful. Some scripts are bundled with Blender, and are discussed later on.

There are dozens of plugins that are also available that extend Blender in two major areas:

- Texture Plugins − change the way a material appears
- Video Sequences − do something neat to a video stream

Click on either of the above hyperlinks to find out more about Plugins.

The default installation of Blender includes several bundled scripts and plugins.

Saving your own Extensions

If you find a script useful for you, download it to your Scripts directory and it will be there when you need it. In a GNU/Linux distribution, the scripts can be found in the folder /usr/share/blender/scripts. In the default Windows installation, scripts are in the C:\Program Files\Blender Foundation\.blender\scripts folder, or wherever the pathspec points to in the User Preferences−>File Paths−>Python field. Once there, change one of your windows to a Scripts Window, and use the Scripts menu to find the script. If it is a non−standard script, you will have to load it using the text editor window and then File−>Run Python Script.

Plugins are saved in the Blender/plugins directory, again located under the Blender install directory. Simply save them, and then, when in the appropriate step in either Textures or Sequnences, simply find them on your hard disk and they will be dynamically loaded for you and begin working immediately.

Previous: Manual/Audio Sequences	Contents	*Next:* Manual/Bundled Scripts

User Manual: Contents | Guidelines | Blender Version **2.45**

There are many scripts that are packaged with Blender, as well as hundreds available for downloading from individual sites. To facilitate good documentation of these resources, an entirely new wiki section has been established. Please refer to that page for more information about bundled and available scripts that extend the functionality of Blender

Python scripts can, after a long and stable history, be incorporated into the Blender menu system and appear as options on a window's menu option. This bundling makes them readily accessible to the user. Currently, there are only three windows that have scripts even as an option:

- User Preferences
- 3D View
- UV/Image Editor

This section provides specific links to further documentation on the specific Python Scripts that are available from Blender menus. For more information on all known scripts, click here for the catalog.

User Preferencess Window

Render

- Backface Cull

Help

- Blender/Python Scripting API − Opens HTML browser to an internet page on Python
- Getting Started − Opens HTML browser to an internet page for newbies
- HotKey and MouseAction Reference − Changes a window to a quick reference
- ID Property Browser − set/inspect properties on any Blender object
- Manual − Opens HTML browser to this wiki
- Release Notes − What's new in this Blender version
- Scripts Help Browser − gives you some info and help on scripts themselves
- Tutorials − points you to tutorials that give step−by−step simple instructions on how to do great things with Blender

Websites

These scripts simply open an HTML page in your HTML browser (Firefox, IE) and display the relevant page at www.blender.org:

- E−Shop: get your Blenderware here!
- Blender Website − www.blender.org
- Developer Community: Programmers developing Blender
- User Community: users helping other users

3D View Window

The available scripts that are shown on the menu vary by the mode you are in. For example, the list of scripts that are useable when editing a mesh in Edit mode are different than the scripts available when working with whole Objects.

Object Mode −> Object −> Scripts

- Apply Deformation
- Axis Orientation Copy
- Batch Object Name Edit
- Bone Weight Copy
- Cookie Cut from View
- Data Copier
- Drop Onto Ground
- Find by Data Use
- Object Name Editor
- Randomize Loc Size Rot
- Selection to DupliGroup
- UV Copy from Active

Edit Mode −> Mesh −> Scripts

- Bevel Center
- Clean Meshes
- Discombobulator
- Edges to Curve
- Mirror Vertex Locations & Weight
- Poly Reducer
- Rvk1 to Rvk2
- Solid Wireframe
- Solidify Selection
- Unfold
- Unweld vertex/ices

Vertex Paint Mode −> Paint

- VCol Gradient
- Self Shadow VCols (AO)
- Copy from Material

Weight Paint Mode −> Paint

- Clean Weight
- Copy Active Group
- Envelope via Group Objects
- Normalize/Scale Weight
- Grow/Shrink Weight
- Weight Gradient

UV/Image Editor Window

Image Mode

These scripts help you when working with images that are probably going to be used as UV−mapped images

- Fix Broken Paths
- Edit Externally
- Consolidate into one image

- Billboard Render on Active

UVs Mode

- Save UV Face Layout
- Seams from Islands

Making a Script 'Bundled'

To make a script registered within blender (to show up in a menu) you need to include a script header. Please see "Registering Scripts" on this page Python API Page.

Additional Resources

Please also see the BlenderArtists Python forum.

Wiki Archive

Remainder of Page is Archived

The remainder of this page is unmaintained and may contain inaccurate or outdated information.
}

Animation

These Scripts are available in the Scripts window, Scripts−>Animation menu.

- armature_symetry.py
- bvh2arm.py
- camera_changer.py
- envelope_assignment.py
- envelope_symmetry.py

Export

These Scripts are available in the File−>Export Menu.

- ac3d_export.py
- bvh_export.py
- DirectX8Exporter.py
- DirectXExporter.py
- lightwave_export.py
- nendo_export.py
- uv_export.py
- videoscape_export.py
- vrml97_export.py
- wings_export.py
- wrl2export.py

- x3d_export.py
- truespace_export.py
- obj_export.py
- off_export.py
- blender2cal3d.py
- radiosity_export.py
- raw_export.py
- save_theme.py

Exporter Feature Matrix

This is a list of support for exporters within Blender.

NA (not applicable), unsupported by the format.
YES/NO or ***** for notes.

Native	Ext	Object	Mesh	Mat	UV	Vert Color	Arm	Animation	Skin	Scene	Lgt	Cam	World
3D Studio	3DS	YES	YES	YES	YES	NO	NO	NO	YES	NA	NO	NO	NA
AC3D	AC	YES	YES	YES	YES	NA	NA	NA	NA	NA	NA	NA	NA
Quake 2	MD2	YES	YES	NA	YES	NA	NA	YES	YES	NA	NA	NA	NA
Wavefront OBJ	OBJ	YES	YES	YES	YES	NA	NA	NA	NA	NA	NA	NA	NA
Native	Ext	Object	Mesh	Mat	UV	Vert Color	Arm	Animation	Skin	Scene	Lgt	Cam	World

FaceSelect

These Scripts are available in 3d window Select menu in Face Select Mode.

- sel_same.py

Help

These Scripts are available in Help menu.

- help_browser.py
- help_getting_started.py
- help_manual.py
- help_py_reference.py
- help_release_notes.py
- help_tutorials.py
- hotkeys.py

Import

These Scripts are available in the File−>Import Menu.

- ac3d_import.py

- bvh_import.py
- lightwave_import.py
- nendo_import.py
- wings_import.py
- truespace_import.py
- off_import.py
- obj_import.py
- paths_import.py
    - SVG
    - AI
    - GIMP
    - EPS
- radiosity_import.py
- raw_import.py
- slp_import.py

Import Feature Matrix

This is a list of support for importers within Blender.

> **NA** (not applicable), unsupported by the format.
> **YES/NO** or * for notes.

Native	Ext	Object	Mesh	Mat	UV	Vert Color	Arm	Animation	Skin	Scene	Lgt	Cam	World
3D Studio	3DS	YES	YES	YES	YES	NO	NO	NO	YES	NA	NO	NO	NA
AC3D	AC	YES	YES	YES	YES	NA	NA	NA	NA	NA	NA	NA	NA
Quake 2	MD2	YES	YES	NA	YES	NA	NA	YES	YES	NA	NA	NA	NA
Wavefront OBJ	OBJ	YES	YES	YES	YES	NA	NA	NA	NA	NA	NA	NA	NA
Native	Ext	Object	Mesh	Mat	UV	Vert Color	Arm	Animation	Skin	Scene	Lgt	Cam	World

Materials

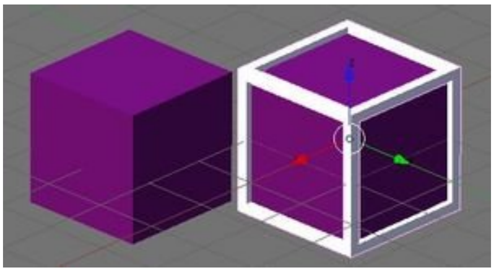
(none available)

Mesh

These Scripts are available in the 3d window Object−>Scripts menu in Mesh Edit Mode.

Apply Deformation
  Apply_def.py
Bevel Center
  Bevels selected mesh edges. bevel_center.py
Clean Mesh
  clean_mesh.py
Cleanup Meshes
Discombobulator
  discombobulator.py

Dispaint
: disp_paint.py



Solid Wireframe
: Creates a new mesh object that extends your mesh geometry to create "solid" edges where vertices connect (an edge). Like reinforcing on a crate.

- fixfromarmature.py
- rvk1_torvk2.py
- skin.py
- unweld.py

Misc

(none available)

Object

These Scripts are available in the 3d window Object−>Scripts menu in Object Mode.

- Axiscopy.py
- batch_name_edit.py
- knife.py
- renameobjectbyblock.py
- obdatacopier.py

Render

(none available)

System

These Scripts are available in the Help−>System menu.

- config.py
- console.py
- doc_browser.py

- sysinfo.py

Themes

These scripts change the color scheme for Blender. If you want them to be your default the next time you load, press Ctrl U and save. To restore to factory defaults, use the User Preferences window, Themes tab to select the Default theme.

UV

These Scripts are available in the UV window, UV menu

- tex2uvbaker.py wiki documented here: Manual/Using_UV_Textures#Texture_Baker

Wizards

These Scripts are available in the Scripts window, Scripts−>Wizards menu.

- BGC − Blender Go Cube − makes 6 images of your 3D space.
- Blender Caliper − measures you meshes and distances in 3D space
- Blender Swarm − makes particles swarm like bees between points
- Klop Utils.py

Help Websites

These Scripts are available in the Help menu.

- Getting Started − opens a browser to the a getting started section of Blender.org
- Hotkey and MouseAction
- Manual − opens the Mediawiki for Blender's User Manual
- Python Scripting
- Release Notes
- Scripts Help − help on the current script running
- Tutorials

These Scripts are available in the Help−>Websites menu.

- E−Shop: get your Blenderware here! (help_web_eshop.py)
- Blender Website − www.blender.org (help_web_blender.py)
- Developer Community − programmers developing Blender (help_web_devcomm.py)
- User Community − users helping other users (help_web_usercomm.py)

Previous: Manual/Extending Blender | Contents | *Next:* Manual/Python Scripting

User Manual: Contents | Guidelines | Blender Version **2.31**

Python Scripting

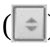
Blender has a very powerful yet often overlooked feature. It exhibits an internal fully fledged Python interpreter. This allows any user to add functionalities by writing a Python script. Python is an interpreted, interactive, object−oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It was expressly designed to be usable as an extension language for applications that need a programmable interface, and this is why Blender uses it. Of the two main ways of extending Blender, the other one being binary plugins, Python scripting is more powerful, versatile yet easier to comprehend and robust. It is generally preferred to use Python scripting than writing a plugin. Actually Python scripting had somewhat limited functionalities up to Blender 2.25, the last of NaN releases. When Open Sourcing Blender many of the new developers gathered around the Foundation elected to work on it and, together with UI change, Python API is probably the single part of Blender which got the greatest development. A full reorganization of what existed was carried out and many new modules added. This evolution is still ongoing and even better integration is expected in forthcoming Blender versions.

The Text Editor

Blender has a Text Window among its windows types accessible via the 

button of the Window Type menu or via SHIFT−F11. The newly opened Text window is grey and empty, with a very simple toolbar (*Text Toolbar*.). From left to right there are the standard Window type selection button and the Window menu. Then the full screen button, followed by two toggle buttons: one which shows/hides the line numbers for the text and one which shows/hides the syntax highlighting. At the end, the regular Menu Button and two settings: the font of the editor, and the number of spaces in a tabulation.

Text Toolbar.

The Menu Button () allows you to select which Text buffer is to be displayed, as well as allowing you to create a new buffer or load a text file. If you choose to load a file the Text Window temporarily becomes a File Selection Window, with the usual functions. Once a text buffer is in the Text window, this behaves as a very simple text editor. Typing on the keyboard produces text in the text buffer. As usual pressing, LMB dragging and releasing LMB selects text. The following keyboard commands apply:

- Alt C or Ctrl C − Copy the marked text into the text clipboard;
- Alt X or Ctrl X − Cut out the marked text into the text clipboard;
- Alt V or Ctrl V − Paste the text from the clipboard to the cursor in the Text Window;

  Blender's cut/copy/paste clipboard is *separate* from the operating system (OS) clipboard. So normally you *cannot* cut/paste/copy out from/into Blender. To access your OS clipboard:

- Shift Ctrl C − To copy text to the OS buffer (e.g. if you want to paste that text into another application);
- Shift Ctrl X − To cut and copy text to the OS buffer;
- Shift Ctrl V − To paste text from the OS buffer (e.g. you copied some text from your web browser or document editor);
- Alt S − Saves the text as a text file, a File Selection Window appears;
- Alt O − Loads a text, a File Selection Window appears;

- Alt F − Pops up the Find toolbox;
- Shift Alt F or RMB − Pops up the File Menu for the Text Window;
- Alt J − Pops up a Num Button where you can specify a linenumber the cursor will jump to;
- Alt P − Executes the text as a Python script;
- Alt U − Undo;
- Alt R − Redo;
- Ctrl R − Reopen (reloads) the current buffer;
- Alt M − Converts the content of the text window into 3D text (max 100 chars).

To delete a text buffer just press the 'X' button next to the buffer's name, just as you do for materials, etc. The most notable keystroke is ALT−P which makes the content of the buffer being parsed by the internal Python interpreter built into Blender. The next section will present an example of Python scripting. Before going on it is worth noticing that Blender comes with only the bare Python interpreter built in, and with a few Blender−specific modules, those described in **REF** .

Other usages for the Text window

The text window is handy also when you want to share your .blend files with the community or with your friends. A Text window can be used to write in a README text explaining the contents of your blender file. Much more handy than having it on a separate application. Be sure to keep it visible when saving! If you are sharing the file with the community and you want to share it under some licence you can write the licence in a text window.

Accessing Python

To have access to the standard Python modules you need a complete working Python install. You can download this from http://www.python.org

Setting the PYTHONPATH environment variable

Be sure to check on http://www.blender.org which is the *exact* Python version which was built into Blender to prevent compatibility issues. Blender must also be made aware of *where* this full Python installation is. This is done by defining a PYTHONPATH environment variable.

Setting PYTHONPATH on Win95,98,Me

Once you have installed Python in, say, C:\PYTHON22you must open the file C:\AUTOEXEC.BATwith your favourite text editor, add a line:

```
SET PYTHONPATH=C:\PYTHON22;C:\PYTHON22\DLLS;C:\PYTHON22\LIB;C:\PYTHON22\LIB\LIB−TK
```

and reboot the system.

Setting PYTHONPATH on WinNT,2000,XP

Once you have installed Python in, say, C:\PYTHON22 Go on the "My Computer" Icon on the desktop, RMB and select Properties. Select the Advanced tab and press the Environment Variables button. Below the System Variables box, (the second box), hit New. If you are not an administrator you might be unable to do that. In this case hit New in the upper box. Now, in the Variable Name box, type PYTHONPATH, in the Variable

Value box,type:

```
C:\PYTHON22;C:\PYTHON22\DLLS;C:\PYTHON22\LIB;C:\PYTHON22\LIB\LIB-TK
```

Hit OK repeatedly to exit from all dialogs. You may or may not have to reboot, depending on the OS.

Setting PYTHONPATH on Linux and other UNIXes

Normally you will have Python already there. if not, install it. You will have to discover where it is. This is easy, just start a Python interactive shell by opening a shell and by typing python in there. Type the following commands:

```
>>> import sys
>>> print sys.path
```

and note down the output, it should look like

```
['', '/usr/local/lib/python2.2', '/usr/local/lib/python2.2 /plat-linux2',
'/usr/local/lib/python2.0/lib-tk', '/usr/local/lib/python2.0/lib-dynload',
'/usr/local/lib/python2.0/site-packages']
```

Add this to your favourite rc file as an environment variable setting. For example, add in your .bashrc the line

```
export PYTHONPATH=/usr/local/lib/python2.2:/usr/local/lib/python2.2/plat-linux2:
                  /usr/local/lib/python2.2/lib-tk:/usr/local/lib/python2.2/lib-dynload:
                  /usr/local/lib/python2.0/site-packages
```

all on a single line. Open a new login shell, or logoff and login again.

A working Python example

Now that you've seen that Blender is extensible via Python scripting and that you've got the basics of script handling and how to run a script, before smashing your brain with the full python API reference let's have a look at a quick working example. We will present a tiny script to produce polygons. This indeed duplicates somewhat the SPACEAdd>>Mesh>>Circle toolbox option, but will create 'filled' polygons, not just the outline. To make the script simple yet complete it will exhibit a Graphical User Interface (GUI) completely written via Blender's API.

Headers, importing modules and globals.

The first 32 lines of code are listed below

Script header

001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032

##
#
Demo Script for Blender 2.3 Guide
#
###S68

```
# This script generates polygons. It is quite useless
# since you can do polygons with ADD−>Mesh−>Circle
# but it is a nice complete script example, and the
# polygons are 'filled'
######################################################

######################################################
# Importing modules
######################################################

import Blender
from Blender import NMesh
from Blender.BGL import *
from Blender.Draw import *

import math
from math import *

# Polygon Parameters
T_NumberOfSides = Create(3)
T_Radius = Create(1.0)

# Events
EVENT_NOEVENT = 1
EVENT_DRAW = 2
EVENT_EXIT = 3
```

After the necessary comments with the description of what the script does there is (lines 016−022) the importing of Python modules. Blender is the main Blender Python API module. NMesh is the module providing access to Blender's meshes, while BGL and Draw give access to the OpenGL constants and functions and to Blender's windowing interface, respectively. The math module is Python's mathematical module, but since both the 'math' and the 'os' modules are built into Blender you don't need a full Python install for this! The polygons are defined via the number of sides they have and their radius. These parameters have values which must be defined by the user via the GUI hence lines (025−026) create two 'generic button' objects, with their default starting value. Finally, the GUI objects works with, and generates, events. Events identifier are integers left to the coder to define. It is usually a good practice to define mnemonic names for events, as is done here in lines (029−031).

Drawing the GUI.

The code responsible for drawing the GUI should reside in a draw function (*GUI drawing*).

GUI drawing

033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057

```
######################################################
# GUI drawing
######################################################
def draw():
        global T_NumberOfSides
        global T_Radius
```

```
        global EVENT_NOEVENT,EVENT_DRAW,EVENT_EXIT

        ########## Titles
        glClear(GL_COLOR_BUFFER_BIT)
        glRasterPos2d(8, 103)
        Text("Demo Polygon Script")

        ######### Parameters GUI Buttons
        glRasterPos2d(8, 83)
        Text("Parameters:")
        T_NumberOfSides = Number("No. of sides: ", EVENT_NOEVENT, 10, 55, 210, 18,
                T_NumberOfSides.val, 3, 20, "Number of sides of out polygon");
        T_Radius = Slider("Radius: ", EVENT_NOEVENT, 10, 35, 210, 18,
                T_Radius.val, 0.001, 20.0, 1, "Radius of the polygon");

        ######### Draw and Exit Buttons
        Button("Draw",EVENT_DRAW , 10, 10, 80, 18)
        Button("Exit",EVENT_EXIT , 140, 10, 80, 18)
```

Lines (037−039) merely grant access to global data. The real interesting stuff starts from lines (042−044). The OpenGL window is initialised, and the current position set to x=8, y=103. The origin of this reference is the lower left corner of the script window. Then the title Demo Polygon Script is printed. A further string is written (lines 047−048), then the input buttons for the parameters are created. The first (lines 049−050) is a Num Button, exactly like those in the various Blender Button Windows. For the meaning of all the parameters please refer to the API reference. Basically there is the button label, the event generated by the button, its location (x,y) and its dimensions (width, height), its value, which is a data belonging to the Button object itself, the minimum and maximum allowable values and a text string which will appear as a help while hovering on the button, as a tooltip. Lines (051−052) defines a Num Button with a slider, with a very similar syntax. Lines (055−056) finally create a Draw button which will create the polygon and an Exit button.

Managing Events.

The GUI is not drawn, and will not work, until a proper event handler is written and registered (*Handling events*).

Handling events

058 059 060 061 062 063 064 065 066 067 068 069 070 071 072 073 074 075

```
def event(evt, val):
        if (evt == QKEY and not val):
                Exit()

def bevent(evt):
        global T_NumberOfSides
        global T_Radius
        global EVENT_NOEVENT,EVENT_DRAW,EVENT_EXIT

        ######### Manages GUI events
        if (evt == EVENT_EXIT):
                Exit()
        elif (evt== EVENT_DRAW):
```

```
        Polygon(T_NumberOfSides.val, T_Radius.val)
        Blender.Redraw()

Register(draw, event, bevent)
```

Lines (058−060) define the keyboard event handler, here responding to the Q with a plain Exit() call. More interesting are lines (062−072), in charge of managing the GUI events. Every time a GUI button is used this function is called, with the event number defined within the button as a parameter. The core of this function is hence a "select" structure executing different codes according to the event number. As a last call, the Register function is invoked. This effectively draws the GUI and starts the event capturing cycle.

Mesh handling

Finally, *Main function* shows the main function, the one creating the polygon. It is a rather simple mesh editing, but shows many important points of the Blender's internal data structure.

Main function

076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096 097 098 099 100 101 102 103 104 105 106 107 108 109

```
######################################################
# Main Body
######################################################
def Polygon(NumberOfSides,Radius):

    ######### Creates a new mesh
    poly = NMesh.GetRaw()

    ######### Populates it of vertices
    for i in range(0,NumberOfSides):
        phi = 3.141592653589 * 2 * i / NumberOfSides
        x = Radius * cos(phi)
        y = Radius * sin(phi)
        z = 0

        v = NMesh.Vert(x,y,z)
        poly.verts.append(v)

    ######### Adds a new vertex to the center
    v = NMesh.Vert(0.,0.,0.)
    poly.verts.append(v)

    ######### Connects the vertices to form faces
    for i in range(0,NumberOfSides):
        f = NMesh.Face()
        f.v.append(poly.verts[i])
        f.v.append(poly.verts[(i+1)%NumberOfSides])
        f.v.append(poly.verts[NumberOfSides])
        poly.faces.append(f)

    ######### Creates a new Object with the new Mesh
```

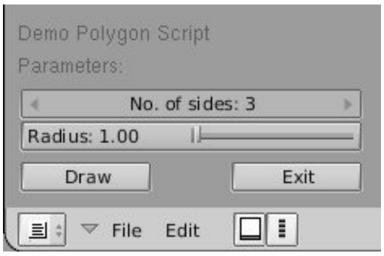
```
polyObj = NMesh.PutRaw(poly)

Blender.Redraw()
```

The first important line here is number (082). Here a new mesh object, poly is created. The mesh object is constituted of a list of vertices and a list of faces, plus some other interesting stuff. For our purposes the vertices and faces lists are what we need. Of course the newly created mesh is empty. The first cycle (lines 085−092) computes the x,y,z location of the NumberOfSides vertices needed to define the polygon. Being a flat figure it is z=0 for all. Line (091) calls the NMesh method Vert to create a new vertex object of co−ordinates (x,y,z). Such an object is then appended (line 092) in the poly Mesh verts list. Finally (lines 095−096) a last vertex is added in the centre. Lines (099−104) now connects these vertices to make faces. It is not required to create all vertices beforehand and then faces. You can safely create a new face as soon as all its vertices are there. Line (100) creates a new face object. A face object has its own list of vertices v (up to 4) defining it. Lines (101−103) appends three vertices to the originally empty f.v list. The vertices are two subsequent vertices of the polygon and the central vertex. These vertices must be taken from the Mesh verts list. Finally line (104) appends the newly created face to the faces list of our poly mesh.

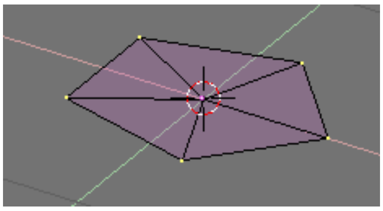
Conclusions

If you create a polygon.py file containing the above described code and load it into a Blender text window, as you learned in the previous section, and press ALT−P in that window to run it, you will see the script disappearing and the window turn grey. In the lower left corner the GUI will be drawn (*The GUI of our example.*).



The GUI of our example.

By selecting, for example, 5 vertices and a radius 0.5, and by pressing the Draw button a pentagon will appear on the xy plane of the 3D window (*The result of our example script.*).



The result of our example script.

Python Reference

The Full Python Application Programmer Interface of Blender has a reference documentation which is a book by itself. For space reason it is not included here. Here it is :

- The Blender Python API Reference (Blender 2.46)

- The Blender Python API Reference (Blender 2.45) out−dated
- The Blender Python API Reference (Blender 2.44) out−dated
- The Blender Python API Reference (Blender 2.43) out−dated
- The Blender Python API Reference (Blender 2.42) out−dated

Python Scripts

There are more than one hundred different scripts for Blender available on the net.

As with plugins, scripts are very dynamic, changing interface, functionalities and web location fairly quickly, so for an updated list and for a live link to them please refer to one of the two main Blender sites, www.blender.org or www.blenderartists.org, and the part of this wiki which is dedicated to them.

Previous: Manual/Bundled Scripts Contents *Next:* Manual/Blender's Plugins System

User Manual: Contents | Guidelines | Blender Version **2.31**

Blender's Plugins System

This section reports an in−depth reference for coding Blender's Texture and Sequence plugins.

Writing a Texture Plugin

In this Section we will write a basic texture plugin and then go through the steps to use a texture plugin. The basics behind a texture plugin is that you are given some inputs; position, and normal values as well as some other info. Then you return intensity, colour and/or normal information depending on the type of texture plugin. All the files necessary to develop plugins as well as a few sample plugins can be found in the blender/plugins. You can alternately get a bunch of plugins from http://www.cs.umn.edu/~mein/blender/plugins Plugins are supported (loaded/called) in Blender using the dlopen() family of calls. For those unfamiliar with the dlopen system it allows a program (Blender) to use a compiled object as if it were part of the program itself, similar to dynamically linked libraries, except the objects to load are determined at runtime. The advantage of using the dlopen system for plugins is that it is very fast to access a function, and there is no overhead in interfacing to the plugin, which is critical when as (in the case of texture plugins) the plugin can be called several million times in a single render. The disadvantage of the system is that the plugin code works just like it is part of Blender itself, if the plugin crashes, Blender crashes. The include files found in the plugin/include/ subdirectory of the Blender installation document the Blender functionality provided to the plugins. This includes the Imbuf library functions for loading and working with images and image buffers, and noise and turbulence functions for consistent texturing.

Specification:

- **#include <plugin.h>**

  Every Blender plugin should include this header file, which contains all of the structures and defines needed to properly work with Blender.

- **char name[]="Tiles";**

  A character string containing the plugin name, this value will be displayed for the texture's title in the Texture Buttons window.

- **#define NR_TYPES 2**, **char stnames[NR_TYPES][16]= {"Square", "Deformed"};**

  Plugins are allowed to have separate subtypes for minor variations on algorithms − for example the default clouds texture in Blender has the "Default" and "Color" subtypes. NR_TYPES should be defined to the number of subtypes required by your plugin, and a name for each subtype should be given (stnames[NR_TYPES]). Every plugin should have at least 1 subtype and a subtype name.

- **VarStruct varstr[]= {...};**

  The varstr contains all of the information Blender needs to display buttons for a plugin. Buttons for plugins can be numerical for input data, or text for comments and other information. Plugins are limited to a maximum of 32 variables. Each VarStruct entry consists of a type, name, range information, and a tool tip.

The type defines the data type for each button entry, and the way to display the button. For number buttons this value should be a combination (ORed) of INT or FLO for the number format, and NUM, NUMSLI, or TOG, for the button type. Text buttons should have a type of LABEL.
The name is what will be displayed on (or beside) the button. This is limited to 15 characters.
The range information consists of three floats that define the default, minimum, and maximum values for the button. For TOG buttons the minimum is set in the pressed state, and the maximum is set in the depressed state.
The tip is a string that will be displayed when the mouse is over this button (if the user has tool tips on). This has a limit of 80 characters, and should be set to the NULL string ("") if unused.

- **typedef struct Cast {...};**

The cast structure is used in calling the doit function, and serves as a way to simply access each plugin's data values. The cast should contain, in order, an integer or float for every button defined in the varstr, including text buttons. Typically these should have the same name as the button for simple reference.

- **float result[8];**

The result array is used to pass information to and receive information from the plugin. The result values are mapped as follows:

Result Index	Significance	Range
result[0]	Intensity value	0.0 to 1.0
result[1]	Red color value	0.0 to 1.0
result[2]	Green color value	0.0 to 1.0
result[3]	Blue color value	0.0 to 1.0
result[4]	Alpha color value	0.0 to 1.0
result[5]	X normal displacement	−1.0 to 1.0
result[6]	Y normal displacement	−1.0 to 1.0
result[7]	Z normal displacement	−1.0 to 1.0

The plugin should always return an intensity value. Returning RGB or a normal are optional, and should be indicated by the doit() return flag "1" (RGB) or "2" (Normal).
Before the plugin is called, Blender includes the current rendering−normal in result[5], result[6] and result[7].

- **float cfra**

The cfra value is set by Blender to the current frame before every render pass. This value is an the frame number +/− .5 depending on the field settings.

- **plugin_tex_doit prototype**

  The plugin_tex_doit function should be prototyped for use by the getinfo function. You do not need to change this line.

- **plugin_tex_getversion**

  This function must be in each plugin for it to be loaded correctly. You should not change this function.

- **plugin_but_changed**

  This function is used to pass information about what buttons the user changes in the interface. Most plugins should not need to use this function, only when the interface allows the user to alter some variable that forces the plugin to do recalculation (a random hash table for example).

- **plugin_init**

  If needed plugins may use this function to initialize internal data. NOTE: This init function can be called multiple times if the same plugin texture is copied. Do not init global data specific to a single instance of a plugin in this function.

- **plugin_getinfo**

  This function is used to communicate information to Blender. You should never need to change it.

- **plugin_tex_doit**

  The doit function is responsible for returning information about the requested pixel to Blender.
  The Arguments
  ◊ **int stype**
  This is the number of the selected subtype, see the **NR_TYPES** and **char stypes** entries above.
  ◊ **Cast *cast**
  The Cast structure which contains the plugin data, see the <emphasis role="bold">Cast</emphasis> entry above.
  ◊ **float *texvec**
  This is a pointer to 3 floats, which are the texture coordinates for which a texture value is to be returned.
  ◊ **float *dxt float *dyt**
  If these pointers are non−NULL they point to two vectors (two arrays of three floats) that define the size of the requested texture value in pixel space. They are only non−NULL when OSA is on, and are used to calculate proper anti aliasing.
  The doit function should fill in the result array and return 0, 1, 2 or 3 depending on what values have been filled in. The doit function should *always* fill in an intensity value. If the function fills in a color value it should return 1, if it fills in a normal value it should return 2, if it fills in everything it should return 3.

Texture/Material Interaction

Blender is somewhat different from most 3D packages in the logical separation between textures and materials. In Blender textures are objects that return certain values, signal generators in fact. Materials control the mapping of textures onto objects, what is affected, how much, in what way, etc. Properly designed plugins

should only include variables to affect the signal returned not the mapping of it. Buttons to control scale, range, axis, etc. are best only included when they make the texture easier to use (in the case of the size button in the Tiles plugin) or they speed up the calculation (the Intensity/Color/Bump subtypes in the Clouds2 plugin). Otherwise the Material Buttons make these buttons redundant, and the interface becomes needlessly complex.

Generic Texture Plugin:

```
#include "plugin.h"

/* Texture name */
char name[24]= "";

#define NR_TYPES 3
char stnames[NR_TYPES][16]= {"Intens","Color", "Bump"};

/* Structure for buttons,
 * butcode name default min max 0
 */
VarStruct varstr[]= {
        { NUM|FLO, "Const 1", 1.7, -1.0, 1.0, ""},
};

typedef struct Cast {
        float a;
} Cast;

float result[8];
float cfra;

int plugin_tex_doit(int, Cast*, float*, float*, float*);

/* Fixed Functions */
int plugin_tex_getversion(void) {
        return B_PLUGIN_VERSION;
}

void plugin_but_changed(int but) { }
void plugin_init(void) { }

void plugin_getinfo(PluginInfo *info) {
        info->name= name;
        info->stypes= NR_TYPES;
        info->nvars= sizeof(varstr)/sizeof(VarStruct);
        info->snames= stnames[0];
        info->result= result;
        info->cfra= &cfra;
        info->varstr= varstr;
        info->init= plugin_init;
        info->tex_doit= (TexDoit) plugin_tex_doit;
        info->callback= plugin_but_changed;
}

int plugin_tex_doit(int stype, Cast *cast, float *texvec, float *dxt, float *dyt) {
        if (stype == 1) {
                return 1;
        } if (stype == 2) {
                return 2;
        }
        return 0;
}
```

Our Modifications:

The first step is to come up with a game plan. What is this plugin going to do, how are the users going to interact with it. For this example we will create a simple texture that creates a simple brick/block pattern. Now we'll copy our generic plugin to cube.c and will fill in the gaps. Its always a good idea to add some comments. First off tell users what the plugin does, where they can get a copy, who they should contact for bugs/improvements, and any licensing restrictions on the code. When using comments make sure you use /* */ style comments. The plugins are in C and some C compilers do not accept // style comments.

```
/*
Description: This plugin is a sample texture plugin that creates a simple
brick/block pattern with it.
It takes two values a brick size, and a mortar size.
The brick size is the size of each brick.
The mortar size is the mortar size in between bricks.
Author: Kent Mein (mein@cs.umn.edu)
Website: http://www.cs.umn.edu/~mein/blender/plugins
Licensing: Public Domain
Last Modified: Tue Oct 21 05:57:13 CDT 2003
*/
```

Next we need to fill in the Name, you should really keep this the same as your .c file, preferably descriptive, less than 23 chars, no spaces, and all lowercase.

```
char name[24]= "cube.c";
```

We are going to keep this plugin simple, and only have one type that deals with intensity. So we need the following:

```
#define NR_TYPES        1
char stnames[NR_TYPES][16]= {"Default"};
```

For our user interface we are going to allow people to change; The size of the brick and mortar, as well as the intensity values returned for the brick and mortar. For that we need to edit the varstr and Cast. The Cast should have a variable for each entry in varstr.

```
/* Structure for buttons,
*  butcode      name           default  min  max  Tool tip
*/
VarStruct varstr[]= {
        {NUM|FLO,   "Brick",      .8,    0.1,   1.0, "Size of Cell"},
        {NUM|FLO,   "Mortar",     .1,    0.0,   0.4, "Size of boarder in cell"},
        {NUM|FLO,   "Brick Int",   1,    0.0,   1.0, "Color of Brick"},
        {NUM|FLO,   "Mortar Int",  0,    0.0,   1.0, "Color of Mortar"},
};

typedef struct Cast {
        float brick,mortar, bricki, mortari;
} Cast;
```

Now we need to fill in plugin_tex_doit, we basically want to break down our texture into "cells" which will consist of a brick and the mortar along the bottom edges of that brick. Then determine if we are in the brick or the mortar. The following code should do that.

```
int plugin_tex_doit(int stype, Cast *cast, float *texvec, float *dxt, float *dyt) {
        int c[3];
        float pos[3], cube;

        /* setup the size of our cell */
        cube = cast->brick + cast->mortar;
```

```
        /* we need to do is determine where we are inside of the current brick. */
        c[0] = (int)(texvec[0] / cube);
        c[1] = (int)(texvec[1] / cube);
        c[2] = (int)(texvec[2] / cube);
        pos[0] = ABS(texvec[0] − (c[0] * cube));
        pos[1] = ABS(texvec[1] − (c[1] * cube));
        pos[2] = ABS(texvec[2] − (c[2] * cube));

        /* Figure out if we are in a mortar position within the brick or not. */
        if ((pos[0] <= cast->mortar) || (pos[1] <= cast->mortar) ||
            (pos[2] <= cast->mortar)) {
                result[0] = cast->mortari;
        } else {
                result[0] = cast->bricki;
        }
        return 0;
}
```

One thing to note, the ABS function is defined in a header in plugins/include. There are some other common functions there as well be sure to take a look at what's there.

Compiling:

bmake is a simple utility (shell script) to aid in the compilation and development of plugins, and can be found in the plugins/ sub−directory of the Blender installation directory. It is invoked by: bmake (plugin_name.c) and will attempt to link the proper libraries and compile the specified C file properly for your system. If you are trying to develop plugins on a windows machine bmake may not work for you in that case you should look into using lcc. You can use the following to compile a plugin with lcc: Assuming you have your plugins in c:\blender\plugins. Here is an example of how you would compile the texture plugin sinus.c Open a dos prompt and do the following: (Note: You'll want to make sure the lcc\bin directory is in your path)

```
cd c:\blender\plugins\texture\sinus
lcc −Ic:\blender\plugins\include sinus.c
lcclnk −DLL sinus.obj c:\blender\plugins\include\tex.def
implib sinus.dll
```

The latest version of lcc does not have implib. Instead, do the following:

```
cd c:\blender\plugins\texture\sinus
lcc −Ic:\blender\plugins\include sinus.c
lcclnk −dll −nounderscores sinus.obj c:\blender\plugins\include\tex.def
```

Note that tex.def is not distributed with the Blender source but is available from the blender plug−in repository at Blender Plug−in Repository. Alternatively, cut and paste the following:

```
EXPORTS
LibMain@12
plugin_but_changed
plugin_getinfo
plugin_init
plugin_tex_doit
plugin_tex_getversion
```

Writing a Sequence Plugin

In this Section we will write a basic sequence plugin and then go through the steps to use a sequence plugin.

The basics behind a sequence plugin are you are given some inputs; 1−3 input image buffers as well as some other information and you output a resulting image buffer. All the files necessary to develop plugins as well as a few sample plugins can be found in the blender/plugins directory. You can alternately get a bunch of plugins from http://www.cs.umn.edu/~mein/blender/plugins

Specification:

- **#include <plugin.h>**

  Every Blender plugin should include this header file, which contains all of the structures and defines needed to properly work with Blender.

- **char name[]="Blur";**

  A character string containing the plugin name, this value will be displayed for the texture's title in the Texture Buttons window.

- **VarStruct varstr[]= {...};**

  The varstr contains all of the information Blender needs to display buttons for a plugin. Buttons for plugins can be numerical for input data, or text for comments and other information. Plugins are limited to a maximum of 32 variables. Each VarStruct entry consists of a type, name, range information, and a tool tip.
  The type defines the data type for each button entry, and the way to display the button. For number buttons this value should be a combination (ORed) of INT or FLO for the number format, and NUM, NUMSLI, or TOG, for the button type. Text buttons should have a type of LABEL.
  The name is what will be displayed on (or beside) the button. This is limited to 15 characters.
  The range information consists of three floats that define the default, minimum, and maximum values for the button. For TOG buttons the minimum is set in the pressed state, and the maximum is set in the depressed state.
  The tip is a string that will be displayed when the mouse is over this button (if the user has tool tips on). This has a limit of 80 characters, and should be set to the NULL string ("") if unused.

- **typedef struct Cast {...};**

  The cast structure is used in calling the doit function, and serves as a way to simply access each plugin's data values. The cast should contain, in order, an integer or float for every button defined in the varstr, including text buttons. Typically these should have the same name as the button for simple reference.

- **float cfra**

  The cfra value is set by Blender to the current frame before every render pass. This value is an the frame number +/− .5 depending on the field settings.

- **plugin_seq_doit prototype**

  The plugin_seq_doit function should be prototyped for use by the getinfo function. You do not need to change this line.

- **plugin_seq_getversion**

  This function must be in each plugin for it to be loaded correctly. You should not change this

function.

- **plugin_but_changed**

  This function is used to pass information about what buttons the user changes in the interface. Most plugins should not need to use this function, only when the interface allows the user to alter some variable that forces the plugin to do recalculation (a random hash table for example).

- **plugin_init**

  If needed plugins may use this function to initialize internal data. NOTE: This init function can be called multiple times if the same plugin sequence is copied. Do not init global data specific to a single instance of a plugin in this function.

- **plugin_getinfo**

  This function is used to communicate information to Blender. You should never need to change it.

- **plugin_seq_doit**

  The sequence doit function is responsible for applying the plugin's effect and copying the final data into the out buffer.
  The Arguments
  - ◊ **Cast *cast**
    The Cast structure which contains the plugin data, see the **Cast** entry above.
  - ◊ **float facf0**
    The value of the plugin's IPO curve for the first field offset. If the user hasn't made an IPO curve this ranges between 0 and 1 for the duration of the plugin.
  - ◊ **float facf1**
    The value of the plugin's IPO curve for the second field offset. If the user hasn't made an IPO curve this ranges between 0 and 1 for the duration of the plugin.
  - ◊ **int x int y**
    The width and height of the image buffers, respectively.
  - ◊ **Imbuf *ibuf1**
    A pointer to the first image buffer the plugin is linked to. This will always be a valid image buffer.
  - ◊ **Imbuf *ibuf2**
    A pointer to the second image buffer the plugin is linked to. Plugins using this buffer should check for a NULL buffer, as the user may not have attached the plugin to two buffers.
  - ◊ **Imbuf *out**
    The image buffer for the plugin's output.
  - ◊ **Imbuf *use**
    A pointer to the third image buffer the plugin is linked to. Plugins using this buffer should check for a NULL buffer, as the user may not have attached the plugin to three buffers.

ImBuf image structure The ImBuf structure always contains 32 bits RGBA pixel data. ImBuf structs are always equal in size, indicated by the passed **x** and **y** value.

User Interaction There is no way for Blender to know how many inputs a plugin expects, so it is possible for a user to attach only one input to a plugin that expects two. For this reason it is important to always check the buffers your plugin uses to make sure they are all valid. Sequence plugins should also include a text label describing the number of inputs required in the buttons interface.

Generic Sequence Plugin:

```
#include "plugin.h"
char name[24]= "";

/* structure for buttons,
 * butcode name default min max 0
 */
VarStruct varstr[]= {
        { LABEL, "In: X strips", 0.0, 0.0, 0.0, ""},
};

/* The cast struct is for input in the main doit function
   Varstr and Cast must have the same variables in the same order */
typedef struct Cast {
        int dummy; /* because of the 'label' button */
} Cast;

/* cfra: the current frame */
float cfra;

void plugin_seq_doit(Cast *, float, float, int, int, ImBuf *, ImBuf *, ImBuf *, ImBuf *);

int plugin_seq_getversion(void) {
        return B_PLUGIN_VERSION;
}

void plugin_but_changed(int but) {
}

void plugin_init() {
}

void plugin_getinfo(PluginInfo *info) {
        info->name= name;
        info->nvars= sizeof(varstr)/sizeof(VarStruct);
        info->cfra= &cfra;
        info->varstr= varstr;
        info->init= plugin_init;
        info->seq_doit= (SeqDoit) plugin_seq_doit;
        info->callback= plugin_but_changed;
}

void plugin_seq_doit(Cast *cast, float facf0, float facf1, int xo, int yo, ImBuf *ibuf1, ImBuf
        char *in1= (char *)ibuf1->rect;
        char *out=(char *)outbuf->rect;
}
```

Our Modifications:

The first step is to come up with a game plan. What is this plugin going to do, how are the users going to interact with it. For this example we will create a simple filter that will have a slider for intensity from 0−255. If any of the R,G, or B components of a pixel in the source image are less then our chosen intensity, it will return black and alpha, otherwise it will return whatever is in the image. Now we'll copy our generic plugin to simpfilt.c and will fill in the gaps. Its always a good idea to add some comments. First off tell users what the plugin does, where they can get a copy, who they should contact for bugs/improvements, and any licensing restrictions on the code. When using comments make sure you use /* */ style comments. The plugins are in C and some C compilers do not accept // style comments.

```
/*
Description: This plugin is a sample sequence plugin that filters out lower
```

```
intensity pixels. It works on one strip as input.
Author: Kent Mein (mein@cs.umn.edu)
Website: http://www.cs.umn.edu/~mein/blender/plugins
Licensing: Public Domain
Last Modified: Sun Sep 7 23:41:35 CDT 2003
*/
```

Next we need to fill in the Name, you should really keep this the same as your .c file. Preferably descriptive, less than 23 chars, no spaces, and all lowercase.

```
char name[24]= "simpfilt.c";
```

The Cast and varstr need to be in sync. We want one slider so we'll do the following:

```
varStruct varstr[]= {
        { LABEL,     "In: 1 strips", 0.0, 0.0, 0.0, ""},
        { NUM|INT,     "Intensity", 10.0, 0.0, 255.0, "Our threshold value"},
};

typedef struct Cast {
        int dummy;                      /* because of the 'label' button */
        int intensity;
} Cast;
```

Now we need to fill in plugin_seq_doit. We basically want to loop through each pixel and if RGB are all less than intensity set the output pixel to: 0,0,0,255 else set it to the input values for that position.

```
int nbrComp = xo * yo * 4; /*There are xo times yo pixels, with 4 composants for each;*/
int i;
char *in1 = (char *)ibuf1->rect; /*ImBuf::rect is an int pointer (containing 4 chars!)
char *out = (char *)outbuf->rect;
for(i=0; i < nbrComp; i+=4)
{
        /* if R and G and B of the pixel are all greater than
           the intensity, keep it unchanged. */
        if((in1[i+0] > cast->intensity) &&
           (in1[i+1] > cast->intensity) &&
           (in1[i+2] > cast->intensity))
        {
                out[i+0] = in1[i+0];
                out[i+1] = in1[i+1];
                out[i+2] = in1[i+2];
                out[i+3] = in1[i+3];
        }
        /* else, make the pixel black and transparent! */
        else
        {
                out[i+0] = out[i+1] = out[i+2] = 0;
                out[i+3] = 255;
        }
}
```

So we wind up with simpfilt.c'

Compiling:

bmake is a simple utility (shell script) to aid in the compilation and development of plugins, and can be found in the plugins/ sub−directory of the Blender installation directory. It is invoked by: bmake (plugin_name.c) and will attempt to link the proper libraries and compile the specified C file properly for your system. If you are trying to develop plugins on a windows machine, bmake may not work for you. In that case you should

look into using lcc. You can use the following to compile a plugin with lcc: Assuming you have your plugins in c:\blender\plugins. Here is an example of how you would compile the sequence plugin sweep.c Open a dos prompt and do the following: (Note: You'll want to make sure the lcc\bin directory is in your path).

```
cd c:\blender\plugins\sequence\sweep
lcc −Ic:\blender\plugins\include sweep.c
lcclnk −DLL sweep.obj c:\blender\plugins\include\seq.def
implib sweep.dll
```

The latest version of lcc does not have implib. Instead, do the following:

```
cd c:\blender\plugins\sequence\sweep
lcc −Ic:\blender\plugins\include sweep.c
lcclnk −dll −nounderscores sweep.obj c:\blender\plugins\include\seq.def
```

Note that seq.def is not distributed with the Blender source but is available from the blender plug−in repository at Blender Plug−in Repository. Alternatively, cut and paste the following:

```
EXPORTS
LibMain@12
plugin_but_changed
plugin_getinfo
plugin_init
plugin_seq_doit
plugin_seq_getversion
```

Previous: Manual/Python Scripting Contents *Next:* Manual/Manual